

DESIGN AND IMPLEMENTATION OF AN INTELLIGENT HEXAPOD

By

OH LAY SHAN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

© Copyright 2006
by
Oh Lay Shan, 2006

CERTIFICATION OF APPROVAL

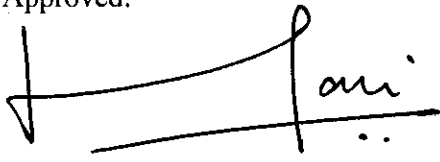
DESIGN AND IMPLEMENTATION OF AN INTELLIGENT HEXAPOD

by

Oh Lay Shan

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

A handwritten signature in black ink, appearing to read 'Haris', is written over a horizontal line.

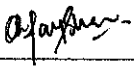
Mr. Mohd. Haris Md. Khir
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

June 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Oh Lay Shan

ABSTRACT

Robots are used to replace humans in some hazardous duty service like bomb disposal and capture information on places that cannot be reached. However, most of the robots used are wheeled robots. Walking machines have the potential to transverse rough terrain that is impossible for standard wheeled vehicles. The mobility of animals, including many insects is typically superior to current legged robots. However, the reality of current technology often encourages engineers to use different designs for the purposes of reducing the number of actuators or simplifying control problem. Thus, the main aim of this project is to design and implement a hexapod walking robot using servo drive mechanism and light weight material for easier control. The project includes constructing the hardware of the machine which is building the structure using appropriate material. The structure is built in the workshop by hand. All sawing and drilling is done using the equipment available in the workshop in Building 22, UTP. To have a stable and rigid structure, the machine is designed to be at its minimum size. This hexapod is designed to move using tripod gait controlled by 3 servo motors. The whole structure is controlled by microcontroller PIC16F877 programmed using C language. The hexapod is able to be controlled manually to move forward, reverse, turning left and right. The walking can be adjusted in 3 different speeds. This hexapod has the object avoidance intelligence using limit switches. The hexapod is able to reverse and find a new walking path if it encounters objects in front of it. With the wireless camera attached in front of the structure, it is able to be used for remote monitoring and rescue operations. The final design is cost effective, light, robust, has easy control and intelligently applicable.

ACKNOWLEDGEMENTS

This project would have been impossible if not been for the help, support and advice from a number of people. Thus, I would like to take this opportunity to express my profound thanks and gratitude to each and every one of them.

First of all, I would like to thank my supervisor, Mr Mohd Haris Md Khir, for agreeing to supervise my final year project. Thanks to his constant guidance and advice, the project is able to amount to success. There were many times where Mr Haris Khir's advice was vital for the continuation of progress. Mr Haris Khir's recommendations and suggestions have been very helpful in inspiring the project.

Secondly, I would like to thank Mr Liao Kuan Lu, a senior who had experience building a hexapod during his final year in University of Adelaide for his advice and guidance throughout the whole development of the project. The help provided from him is very helpful and important for the progress of the development.

Thirdly, I would like to express gratitude to my course mates for helping me out in buying materials and providing transportation to get things done regarding the project. They are namely Miss Poo Hwei Nee, Miss Teo Lee Na and also Mr. Tong Kin Wah. Besides that, their moral support and endless care and valuable opinions and knowledge towards the development of the structure are very much appreciated.

Last but not least, I would also like to thank other individuals who have been both directly and indirectly helpful, especially the lab technicians, Mr Isnani, Mr Musa and Ms. Siti Fatimah for opening the labs and providing supplies from the stores.

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ABBREVIATIONS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Background Studies.....	1
1.2 Problem Statement	2
1.3 Objectives and Scope of Studies	3
CHAPTER 2 LITERATURE REVIEW AND THEORY	4
2.1 Development of Walkers.....	4
2.2 Material Selection	6
2.3 Motors and Drives.....	7
2.4 Walking Gaits.....	8
2.5 Intelligence and Application	10
CHAPTER 3 METHODOLOGY / PROJECT WORK	12
3.1 Methodology	12
3.2 Requirements.....	14
3.2.1 Hardware Requirements	14
3.2.2 Software Requirements.....	16
CHAPTER 4 RESULTS AND DISCUSSION	17
4.1 Mechanical Construction.....	17
4.2 Circuit Design	24
4.3 R/C Servo Motor	25
4.4 Walking Algorithm Design	26
4.4.1 Forward and Reverse Walking Sequence	27
4.4.2 Rotating Left and Right Walking Sequence	30
4.5 PIC Programming.....	32
4.5.1 Control Optimization	39
4.5.2 Speed Control Analysis	42
4.6 Shortcomings, Challenges and Obstacles.....	44
CHAPTER 5 CONCLUSION.....	45
REFERENCES.....	46

APPENDICES	47
Appendix A LINE TRACKING CIRCUIT	48
Appendix B GANTT CHART	49
Appendix C PARALLAX SERVO MOTOR.....	50
Appendix D 16F877 DATA SHEET	52
Appendix E MAIN CONTROLLER BOARD CIRCUIT.....	53
Appendix F 'MAIN.C' SOURCE CODES	54
Appendix G 'MANUAL.C' SOURCE CODE	55
Appendix H WALKING MOTION C CODES	57
Appendix I 'AVOID.C' C CODES	60
Appendix J 'LCD.C' C CODES	62
Appendix K 'KEYPAD.C' C CODES.....	66

LIST OF TABLES

Table 1 : Hexapod walking gait using tripod technology.	9
Table 2 : Hardware Requirements.....	15
Table 3 : Software Requirement.	16
Table 4 : Measurement of structure parts.....	22
Table 5 : Forward walking sequence with pulse-width value.....	29
Table 6 : Reverse walking sequence with pulse-width value.	29
Table 7 : Pulse-width values for left-turn walking sequence.....	31
Table 8 : Pulse-width values for right-turn walking sequence.....	32
Table 9 : Object location based on the input from limit switches.....	37
Table 10 : Robot behavior based on the object location.	39
Table 11 : Pulse-width values for forward walking sequence.	40
Table 12 : Pulse-width values for reverse walking sequence.	41

LIST OF FIGURES

Figure 1 : Wheeled bomb disposal robots.....	2
Figure 2 : Robot III controlled using actuators and drives.....	3
Figure 3 : General Electric Walking Truck.....	4
Figure 4 : NASA Walking Beam.	5
Figure 5 : Hexapods – RHex.....	6
Figure 6 : Duration Pulses to control Servo Motor.	8
Figure 7 : Alternating Tripod Gait [12].	9
Figure 8 : Line Tracking Sensor.....	10
Figure 9 : Ultrasonic range finder.	11
Figure 10 : Flow chart of project development.....	13
Figure 11 : Mechanical structure construction stages.....	17
Figure 12 : Cut and drilled aluminum for body chassis.	18
Figure 13 : Parts placement for body chassis.....	18
Figure 14 : Cut and drilled middle leg servo mounts.....	19
Figure 15 : Middle leg servo mounts and standoffs attached to the chassis.	19
Figure 16 : Cut and drilled upper and lower leg pieces.	20
Figure 17 : Middle leg cut and drilled.....	20
Figure 18 : Mechanical linkages.	20
Figure 19 : Assembled left and right leg.....	21
Figure 20 : Assembled middle leg.	21
Figure 21 : Complete assembled structure.....	21
Figure 22 : Limit switches attached to servo motors.	23
Figure 23 : Final mechanical structure.....	23
Figure 24 : Main controller board.....	24
Figure 25 : Schematic for main controller board.	25
Figure 26 : Duration of PWM generated to control servo motor.	26
Figure 27 : Leg position sequence to achieve forward walking.....	28
Figure 28 : PWM signals generated for forward walking sequence.	30
Figure 29 : Robot leg positions for turning sequence.	31
Figure 30 : PWM signals generated for turn left walking sequence.	32
Figure 31 : Flow diagram of MAIN Loop.	33

Figure 32 : Flow diagram for MANUAL loop.....	34
Figure 33 : PWM generated to servo motor during forward locomotion.	36
Figure 34 : PWM generated to servo motor during left turning locomotion.	36
Figure 35 : Flow diagram of AVOID loop.....	37
Figure 36 : Input signals from limit switches during AUTO mode.	38
Figure 37 : Simplified sequence to achieve forward walking.....	40
Figure 38 : PWM generated for forward/reverse motion using 4 frames/cycle.....	41
Figure 39 : Signals generated for HIGH speed forward/reverse locomotion.....	42
Figure 40 : Signals generated for MEDIUM speed forward/reverse locomotion.....	43
Figure 41 : Signals generated for LOW speed forward/reverse locomotion.	43

LIST OF ABBREVIATIONS

AC	Alternating Current
DC	Direct Current
IR	Infra Red
PCB	Printed Circuit Board
PIC	Programmable Interrupt Controller
PWM	Pulse Wave Modulation
Servo	Servo motor

CHAPTER 1

INTRODUCTION

1.1 Background Studies

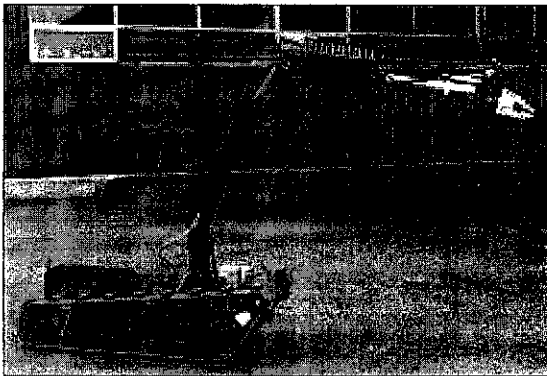
Despite massive deforestation, over 50% of the earth's land mass is still impassable by wheeled or tracked vehicles. Thus, walking machines are much more efficient for handling rough terrain, soft surfaces, and climbing. Walkers are a class of robots that imitate the locomotion of animals and insects. Essentially, walker robots use legs for locomotion which is already hundred of million years old. Sophisticated walkers imitate insects, crabs and sometimes humans.

The main contributor to the birth of walking machines was the internal combustion machine, which made all manners of new vehicles possible [1]. Robot insects, or hexapods, were not commonly experimented with because of the control challenges associated with the limbs. However, the computer control research in 1970's has improved techniques in controlling and can be applied in small machines. Stored-program control is needed because it provides the flexibility to modify the coordination of the various legs to achieve a stable walking gait [2]. Walking robots are one of the most interesting results of today's advanced technology.

The first walking machines in history were mechanical toys normally operated from a rotary power source such as windup spring and clockwork, drive cranks or cams to which the legs were attached. These toys could be considered the proof of concept devices that instilled the idea that larger walking machines could be possible. The earliest attempt to construct a walking machine was made in Britain in 1940 by A.C Hutchison and F.S Smith. [3] They built a machine with four independently controlled legs that walked using the quadruped crawl gait.

1.2 Problem Statement

Walkers have a better degree of mobility especially in environment with obstacles or danger to human. Without risking life or limb, robots can replace humans in some hazardous duty service like bomb disposal and capture information on places that cannot be reached by human. Similar robots can venture into dangerously polluted environments, such as chemical spills and radioactive "hot zones" in nuclear power plants which unprotected human would die quickly. 'MERV' and Mini Andros II as shown in **Figure 1** are examples of robots used for the similar purposes. Both are used for bomb-disposal and for bringing video cameras and microphones into dangerous areas, where a human might get hurt or killed. However, both are wheeled vehicles which can only travel on an even terrain.



MERV, Bomb Disposal Robot[4]



Mini Andros II[5]

Figure 1 : Wheeled bomb disposal robots.

In building a walking machine, the biggest problem most commonly faced is the weight to power problem and of course the extremely complicated controls [6]. Many devices cannot even support themselves, let alone lift a leg to take a step. In order to compensate the weight problem, many hexapods are built using actuators and drives which are very costly. Robot III, the six legged robotic insect as shown in **Figure 2**, designed by Roger Quinn and Roy Ritzmann is controlled pneumatically by actuators. Mechanical design is very crucial too in a hexapod design as good programming can never make up for bad mechanical design.

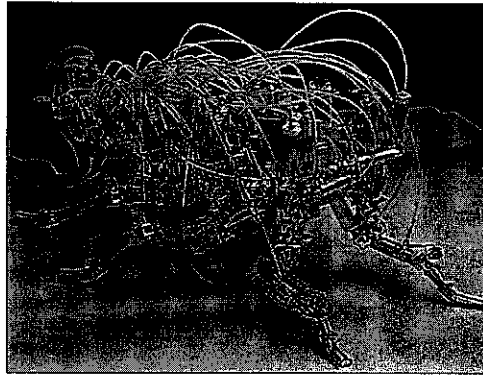


Figure 2 : Robot III controlled using actuators and drives.

The mobility of animals, including many insects is typically superior to current legged robots. This fact recommends the use of animal designs in robots. However, the reality of current technology often encourages engineers to use different designs for the purposes of reducing the number of actuators or simplifying control problem.[7] Thus, to build an effective and simple hexapod, the alternating tripod gait technology can be applied.

1.3 Objectives and Scope of Studies

The main aim of this project is to design and implement a six legged robot which use a tripod technology for its locomotion and has object avoidance ability. This project has to achieve the following design parameters upon completion:

- Light weight, robust and stable structure.
- Cost effective, using only 3 servo motors to control 6 legs.
- Implementing tripod technology in the walking algorithm.
- Able to move forward, backward, turn left and turns right.
- 3 different speed of locomotion.
- Multi functional controller board.
- Ability to avoid object and find new walking path.
- Remote monitoring or surveillance.

This whole project includes constructing a robust and stable structure, circuit design, algorithm design, PIC programming and optimization of controls. The robot is completely built and functioning well in about 16 weeks time.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 Development of Walkers

To be correctly termed a robot, a machine must have these basic components or functions, that is, inputs, in the forms of sensors and detectors, or stored instructions and programs some extent of intelligence, to interpret, respond and react towards its inputs; and actuators, which are controlled directly by the intelligence [8], whether in forms of motors or other output devices (tone, synthesizer). Therefore, it can be concluded that a robot is a machine, assigned and designed to perform or carry out a task, guided by the information from its program or sensors.

In 1940's an attempt was made by A.C Hutchison and F.S. Smith to build the first walking machine that can travel on rough terrain. The device used a pair of hydraulic actuators with a rolling thigh joint, which acted as a kind of inverted wheel, and a telescoping leg. The control mechanism was a series of levers connected to the hands and legs of the operator in a feedback loop. The same concept was used by General Electric for its 'Walking Truck' as shown in **Figure 3** designed by R.S Mosher.

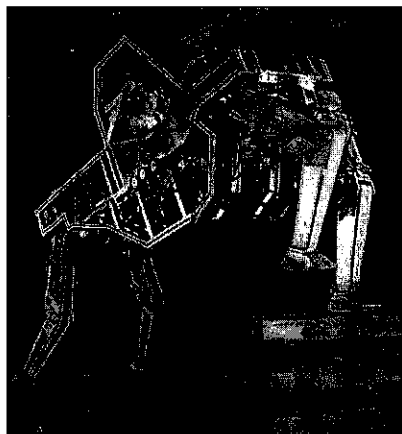


Figure 3 : General Electric Walking Truck.

The machine is a quadruped and was servo-controlled by human driver. The front legs of the machine were controlled by human's arms and back legs followed the movement of the operator's own legs. The machine weighs 1400kg and the hydraulic power was generated by pumps connected to 90 horsepower engine. Although it functions well, it is very demanding on the driver.



The next series of walking machine developments, NASA started to research on the problems of mobility on the moon and other planets. The Walking Beam Machine as shown in **Figure 4** was developed by Wendell Chun of Martin Marietta Astronautics in 1989. The machine can negotiate variety of rough terrains, drill core samples for analysis. The simplified version which measures only $\frac{1}{4}$ of the working model weighs 163kg and is powered by 24V dc source.

Figure 4 : NASA Walking Beam.

Many researches and projects have been carried out recent years to develop hexapods that have mobility increasingly more similar to that of cockroach. Many recent studies on cockroaches propose a new approach to build walking robots that use insects as a source of inspiration to improve the speed and all-terrain performances. Robots like *Sprawlita* , *Whegs I & II* and *RHex* show that rather than just copying the morphology of a cockroach, the solution is to better understand the way it dynamically walks and to implement the fundamental characteristics that explain its performances in a functional and simple robot [9].

The design and control of RHex shown in **Figure 5** was inspired by recent research in biology in particular cockroach locomotion. The RHex research group (at McGill, UC Berkeley, U. Michigan, and recently Carnegie Mellon University) has successfully captured some of the key biomimetic functions in the simple RHex morphology. This has imbued RHex with outstanding mobility over many types of terrain. Each leg has,

again, only one actuator located at the hip and rotates in the sagittal plane. A well chosen visco-elastic structure provides compliance which is useful to maintain a good contact with the ground and to absorb the foot impact without any active control.

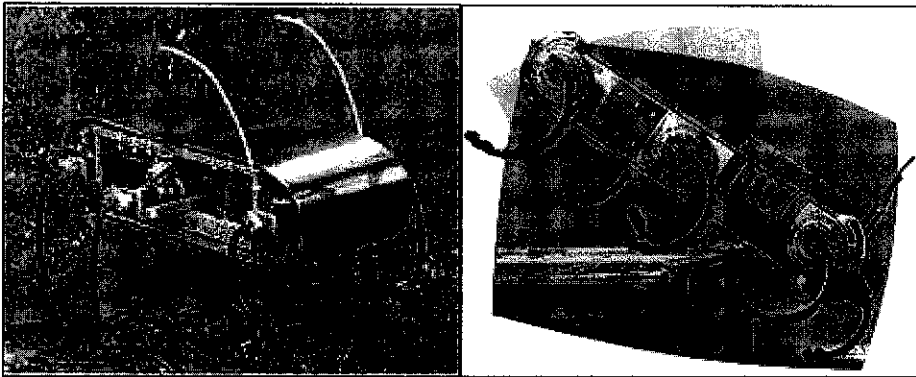


Figure 5 : Hexapods – RHex.

Although with higher efficiency and better mobility, the controls are definitely more complex. Most hexapods are built by using actuators and motors to achieve more degree of freedom in the structure. And this in turn causes the construction to be more costly and adding more weight to the structure. Actuators are typically heavy and reducing their number can increase the payload or range of the robot. A more efficient way of constructing the same hexapod will be by controlling the locomotion electronically. This is when the tripod technology can be implemented. Only 3 servo motors are needed and the chassis of the structure can be built by using scrap aluminums. This not only simplifies the whole design but also cut down the cost on building a walker.

2.2 Material Selection

Most robots use high tensile steel material for its chassis, since they are more suited for heavy duty tasks, such as lifting, punching, and maneuvering heavy equipments. However, the preferred material for experimental robots remains focused on the lighter materials, especially in the case of a walking robot. The structure is preferably light weight, so that it will not impose higher stress or load on the rest of the chassis, and also, the motors, and prime mover. Other than that, it also helps the running and operation of the hexapod to be more efficient and more capable in terms of flexibilities and manageable. The center of gravity of the structure will have to be maintained at a suitably low position too, to increase stability while moving.

The best material to be used is aluminum, because it is strong for its light weight, and its nice metallic sheen, for aesthetical purposes. Steel is also sometime used for stronger support. However it is not very much preferred because it is quite heavy, clumsy and hard to maneuver. Wood is also good for light support, coverage, and finish, because is light weight, though relatively weak for its weight which does not meet the strength requirement of the structure. Another reason for using aluminum is because it is reasonably soft, which means it is easily workable or highly machinable. Therefore, it is easy to shape, drill and cut aluminum bars into the parts needed to assemble into the hexapod structure.

2.3 Motors and Drives

Most of the hexapod design studied earlier used DC motors or actuators to control the movement of the hexapod. Normally the motors and actuators are used in heavy duty tasks machine which require higher torque and power. As for hexapod design especially a small scale experimental robot, such motors are not needed. Hexapod's movements are best to be controlled in positioning and rotation in specific angle. This can be achieve by servo motors which is light, easier control and cheaper.

Servo motors are able to rotate accurately between 180 degrees to a maximum of 360 degrees. Servo motor can get to hold a position, and then continually pulsing it to maintain the desired fixed position. Normally this move-hold application is done by hydraulics to provide smooth move hold operation. However, hydraulics is for big scale machine control, which involves heavy and high powered machine. Thus, it is not applicable for hexapod.

A servo motor is considered to be the most ideal of the other motors, because it has a built-in speed reduction gear box unit, which provides all the torque magnification and speed reduction. Moreover, the servo motor is capable of achieving a particular angle at any one time, which is just what the hexapod needs when it is raising legs for locomotion. The desired servo shaft orientation is achieved by sending in appropriate pulse widths into the servo motor pulse input terminal, which means, the servo interprets PWM to control its shaft orientation.

The parameters for pulse are that it has a minimum width, a maximum width and a repetition rate. Referring to **Figure 6**, the convention is that a pulse of approximately

1.5ms is the neutral point for the servo. Neutral is defined to be the position where the servo has exactly the same amount of potential rotation in the counter clockwise direction as it does in the clockwise direction. When the pulse sent to a servo is less than 1.5ms, the servo positions and holds its output shaft some numbers of degrees counter clockwise from the neutral point. When the pulse is wider than 1.5ms, the opposite occurs. Generally, the minimal pulse will be about 1ms and maximal will be 2ms. The amount of power applied to the motor is proportional to the distance it needs to travel. If the shaft needs to turn a large distance, motor will run at full speed else it will run in lower speed. This is called proportional control.

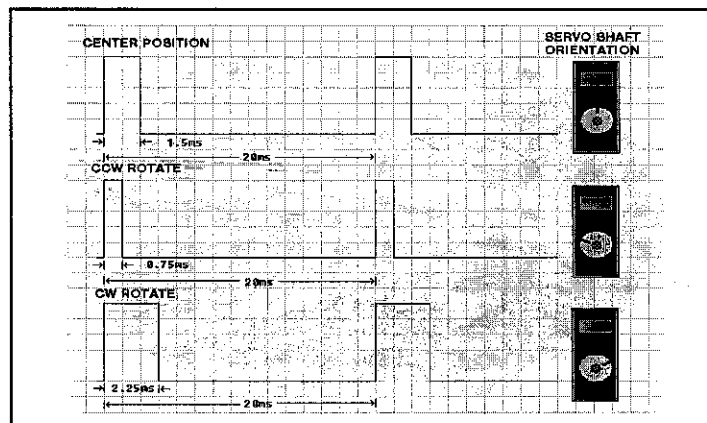


Figure 6 : Duration Pulses to control Servo Motor.

2.4 Walking Gaits

One of the main objectives with legged locomotion is to achieve stability. All legged animals and machines face the potentially dangerous problem of falling over. A six legged robot is stable when at least three of its legs are touching the ground, forming a tripod [10]. During locomotion, the simplest form of stability is to pass without a break from one stable state to another. Most walking machines pass through stages in the locomotion cycle which are not stable, and the machines fall temporarily. If the gait does not contain some phases which are stable, the machine will not be able to stop moving without falling over unless it changes its gait. Most insects change from stable gait at low speed to more unstable gaits at higher speed where balance comes into play [11]. Since it is difficult to achieve dynamic balance with a walking machine, most walking machines are designed to balance statically. The most

common way to do this is to use six legs and move them in triples so that three legs always support the robot.

Figure 7 shows the normal tripod gait used to control a six legged robot. The ovals indicate a foot's position. Black means the foot is in contact with the ground/surface. Gray indicates that the foot is suspended in the air for that step.

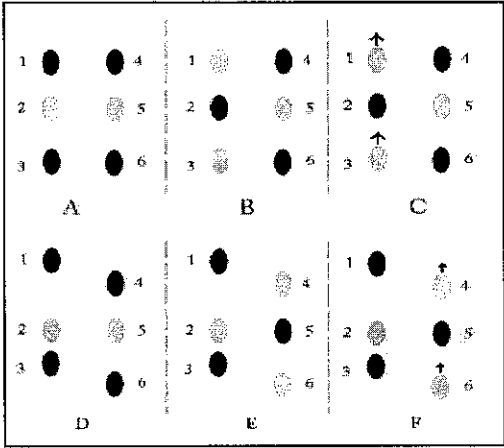


Figure 7 : Alternating Tripod Gait [12].

Referring to **Figure 7**, **Table 1** shows the explanations on how the tripod technology can be implemented in a hexapod walking gait.

Table 1 : Hexapod walking gait using tripod technology.

Position	Explanation
A	Starting position. Center leg is centered so that foot 2 and 5 are suspended in air
B	Center leg swivels so that foot 2 is in contact, raising the left side of the robot. Feet 1 and 3 are suspended in air.
C	Feet 1 and 3 are moved forward by rotating Left/Right motors to forward position, feet 1/3 still suspended.
D	Center leg is centered so that foot 2 and 5 are suspended in air.

E	Center leg tilts so that foot 5 is down, 2 is suspended. Feet 4/6 are suspended.
F	Feet 4/6 are moved forward by rotating Left/Right motors to forward position, feet 4/6 still suspended

2.5 Intelligence and Application

Many hexapods are installed with at least one type of intelligence to add on the practical value. Since the experimental structure is small, it cannot be designed to lift weight or carry things. However, it can be designed to either sense object or have collision avoidance capability or even line tracking ability. Line tracking ability is currently the most famous or rather highly applied technology in robotics nowadays. This technology is simple yet powerful as it will help the robot to move according to lines without human interference, which means automatic.

The line tracking sensor shown in **Figure 8** can be mounted to the underside of a robot chassis toward the front of the structure. The sensor will operate in an extremely wide range from about 0.5” from the floor to almost touching the surface. An infrared LED is paired with an infrared detector. The LED illuminated and directed to the surface where the line is to be detected. The detector is biased on and fed into a comparator to clean up the signal. Refer to **Appendix A** for the circuit. If the sensor sees the white line on a black surface, the output will be low otherwise the output will be high.

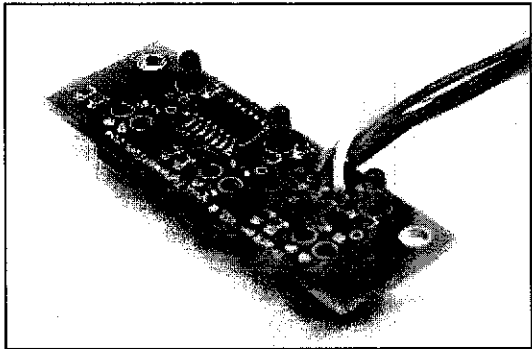


Figure 8 : Line Tracking Sensor.

Ultrasonic range finder as shown in **Figure 9** is normally used in robotics either for room mapping or object avoidance ability [13]. This ultrasonic range finder is a small printed circuit board that measures 1 ¾ X ¾ inches with two ultrasonic transducers mounted in front. The ranger requires a 5V power supply capable of handling roughly 50 mA of continuous output. One transducer is used to send an ultrasonic sensor and the other transducer receives the signal reflection.

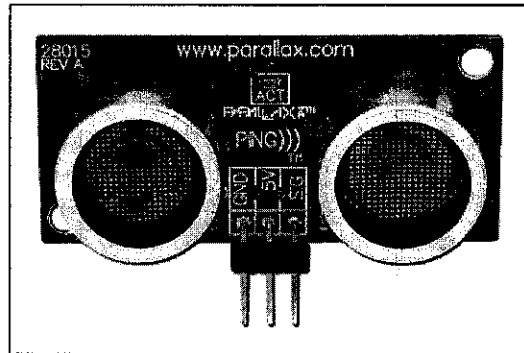


Figure 9 : Ultrasonic range finder.

This ultrasonic range finder works by sending a pulse of sound outside the range of human hearing. This pulse travels at the speed of sound (1.1 ft/millisecond) away from the ranger in a cone shape. If any objects are in the path of the pulse, the sound is reflected off the object and back to the ranger. The ranger is paused for a brief interval after the sound is transmitted, and then awaits the reflected sound in the form of echo. The controller driving the ranger requests the device to create a 40-kHz sound pulse and then waits for the return echo. If echo is received, the ranger reports this echo to the controller and then the distance to the object can be computed based on the time elapsed.

CHAPTER 3

METHODOLOGY / PROJECT WORK

3.1 Methodology

The whole project is undertaken in 1 semester's time. Project development includes researches, mechanical construction, controller design and implementation, walking algorithm design, PIC programming and optimization. Researches are done on the mechanical consideration, materials selection, balancing structure as well as walking algorithm and also object avoidance ability.

The development stages of the whole project are planned out even before the research is carried out. The period of time available to design and implement the project is only about 4 months long. The first two weeks are allocated for researching on all details and information needed. The project work started when construction of the mechanical parts of the hexapod took place. The construction took about 2 week which includes designing, constructing, stabilizing and finalizing. When the structure is finalized, circuit design is carried out which took up 2 weeks to fabricate.

The following stage is more on software development. Walking algorithm is designed and all steps are planned out on how the programming should be done. Chip programming to achieve specific movements and controls can consume a lot of time. About 3 weeks is taken to familiarize with the software and develop the programs. This includes troubleshooting and also fine tuning the software to achieve the best and most efficient controls. The final stage is modification for controls optimization as well as intelligence implementation. **Figure 10** shows the development of the project stage by stage. Refer to **Appendix B** for Gantt Chart.

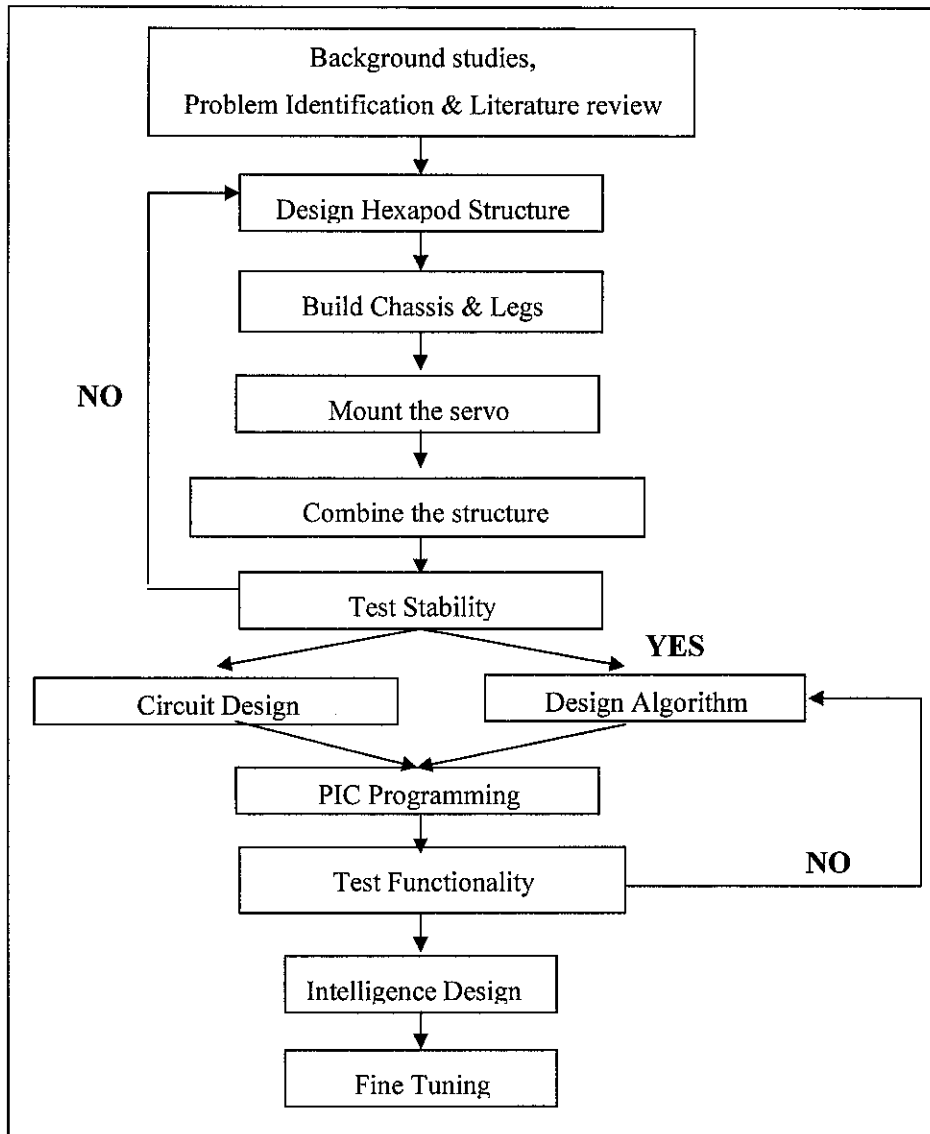


Figure 10 : Flow chart of project development.

The first stage of development is researching on background studies, problem identification and also literature review. With all information needed in hand, the mechanical structure is designed to the best weight and size. The construction of the structure consists of few stages. First, the chassis and legs are constructed according to the size determined earlier. Then servo motor is mounted on the appropriate placing. Lastly, the front and back legs, the middle leg attaching to the servo motor and the chassis is combined to form the final structure. It is then tested for stability before the next stage of development is carried out. If the structure is not stable and strong, the development of the project will then be back tracked to designing the hexapod structure again and each step is repeated until a stable and desired structure is obtained. The construction of this hexapod will be further discussed in Section 4.1.

When the structure is finalized and stable, controller's circuit and structure's walking algorithm is designed. Controller's circuit design involves planning out the devices to be used, schematic design and also generating printed circuit board layout. Components will then be soldered on to the PCB. Walking algorithm is designed using tripod technology or tripod gait. After determining the walking pattern and having a functioning controller board, programming is done. Programming is done using C language and compiled into HEX files to be uploaded into the microcontroller PIC16F877.

Functionality test will be conducted by connecting the servo motors on the structure to the controller board and observing the walking pattern. If the movement is not as desired, the walking algorithm will be redesigned and programming has to be modified. This process is repeated until the structure functions according to algorithm and programming done earlier. When the structure has achieved its best walking gait, intelligence is added into the hexapod.

Since this is walking hexapod where the whole structure is moving up and down, line tracking sensor is not suitable to be mount on the structure. Thus, object avoidance ability is applied as intelligence to this hexapod. Instead of using ultrasonic range finder which is more costly and more complicated, simple limit switches are attached to the front of the structure to enable it to sense the present of obstacles and finding new path. The final stage of the project will be fine tuning and finalizing. This includes optimization of the structure and analysis on the control efficiency.

3.2 Requirements

The progress of the project has to be supported by facilities, equipments, components and tools which can be categorized into hardware requirements and software requirements as listed in **Table 2** and **Table 3**.

3.2.1 Hardware Requirements

During the construction phase of building the robot, there are a number of tools and equipments that will be required. As for the circuit design, there are also various components and electrical equipment needed. Hardware and their application are listed in **Table 2**.

Table 2 : Hardware Requirements.

Hardware	Applications
Servo Motors	There will be a total of 3 Standard PARALLAX Servo motors used in the design. Two for the leg controls and another one will be used to control the middle leg balancing. Refer to Appendix C for Servo Motor's Specification Sheet.
PIC Microcontroller	Controls of the locomotion will be done by electronic circuit board controlling the servo motors. PIC 16F877 will be used as the main microcontroller. Please refer to Appendix D for 16F877 data sheet.
Electronic Components	All electronics components are obtained from the store in Building 22. Such components are needed to construct the programming and target board. Components taken are like PIC 16F877, 4MHz crystal oscillators, voltage regulator, 1K Ω resistors, push buttons, dip switches, male and female connectors, buzzer, LEDs, and toggle switches.
Electrical Equipments	Electrical equipment used for the development are like power supply unit, digital multimeter, dual channel digital oscilloscope and also soldering gun.
Tools and Equipments	Hardware tools used are like the hacksaw, work bench vise, hand held drill and drill bit sets, Stationary drill and drill bit sets, Various clamps, Pro's Kit assorted tool set – pliers, screwdrivers, spanner, cutters, Metal files and hack saw, Soldering iron and fluxed solder lead, with sucker apparatus. Others are like various pliers, wrench and screwdrivers as well as wire strippers, cutters, solder and utility knife.

3.2.2 *Software Requirements*

There are a few types of software used during the programming phase. **Table 3** shows the types of software used and their application.

Table 3 : Software Requirement.

Software	Applications
PIC C Compiler	All software programming are to be written and compiled on PIC C COMPILER using C Language. This software will generate the HEX files to be programmed into the particular PIC.
BumbleBee	Bumblebee is free software downloaded from the internet which allows on board programming. It transfer the HEX codes into the 16F877 PIC on the target board through a programmer board without removing the PIC microcontroller.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Mechanical Construction

The six legged robot is built in evolutionary stages as construction progresses. Each development can be thought of as an evolutionary step in the process of creating an insect-like artificial lifeform. The mechanical construction stage is a genesis stage where the robot's body is constructed. The construction involves the robot's aluminum chassis, legs, servo mounts and mechanical linkages. **Figure 11** shows the construction stages of the structure.

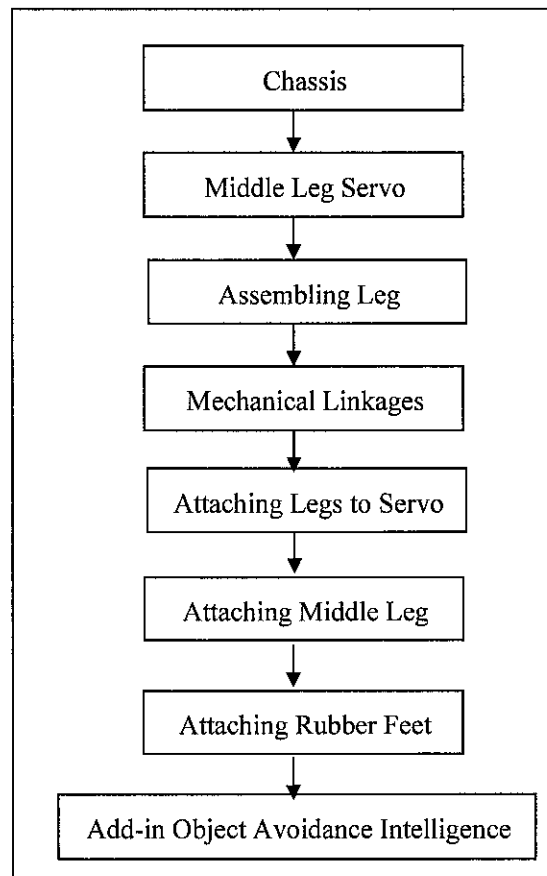


Figure 11 : Mechanical structure construction stages.

The chassis that is built will carry the servo motors and the legs. 5 pieces of aluminum that will make up the body in which three servo motors will eventually be mounted are cut and drilled. Material used for the body chassis is the L-shaped 12mm wide aluminum. **Figure 12** shows the cut and drilled aluminum for the body chassis.

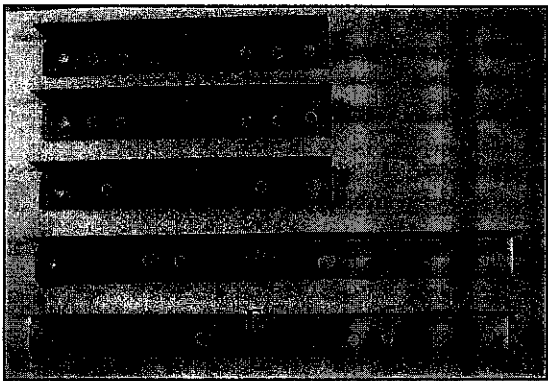


Figure 12 : Cut and drilled aluminum for body chassis.

To assemble the individual pieces, place part D and Part E on a flat surface and lay pieces A, B and C on top, aligning the holes. Two servo motors are attached to Part A and B as shown in **Figure 13**.

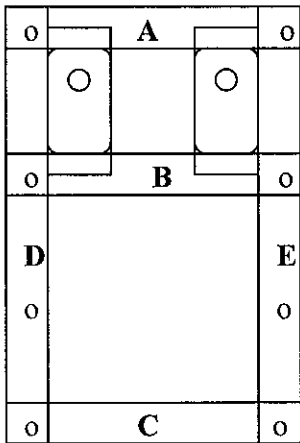


Figure 13 : Parts placement for body chassis.

The next subassembly is the servo mounts for the middle leg. **Figure 14** shows 2 small L-shaped aluminum pieces are cut and drilled to form the middle leg servo mounts.

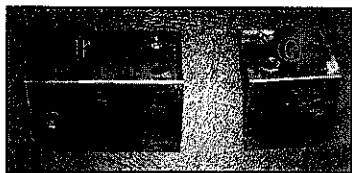


Figure 14 : Cut and drilled middle leg servo mounts.

Servo mounts are then attached to the servo using screws, washers and nuts. Piece F is used to be attached to the side of the servo closest to the servo shaft while piece G is used to be attached to the other side. Next, the chassis is assembled together as shown in **Figure 15**.

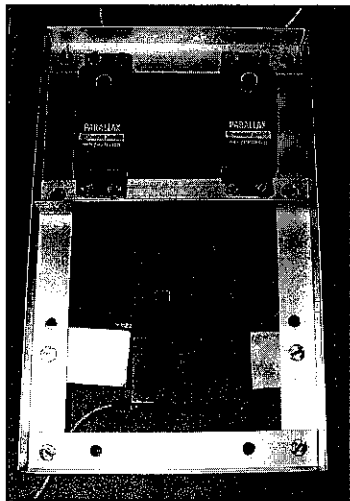


Figure 15 : Middle leg servo mounts and standoffs attached to the chassis.

The next part will be constructing the legs. The upper parts of the legs are built using the L-shaped 12mm aluminum and the lower part are built using the L-shaped 9mm wide aluminum. As for the middle legs attaching to the servo blade, a flat scrap printed circuit board is used to build it. **Figure 16** shows the cut and drilled upper and lower leg pieces. **Figure 17** shows the material used for middle leg construction.

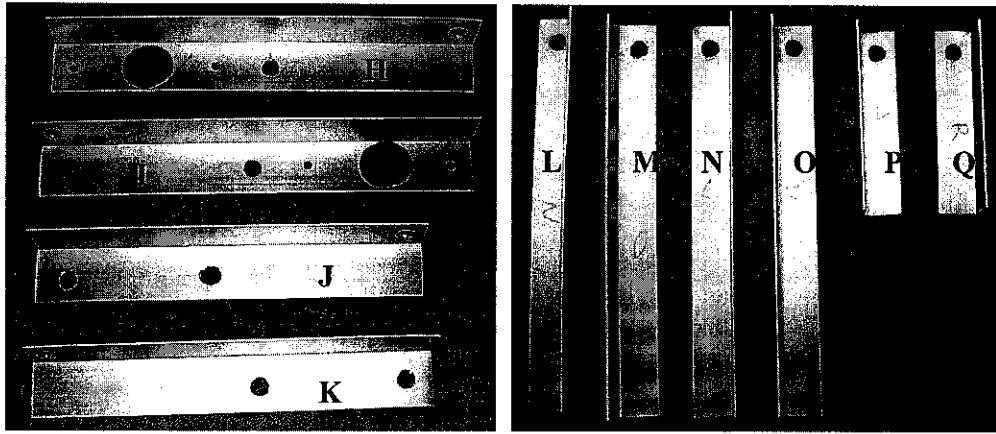


Figure 16 : Cut and drilled upper and lower leg pieces.



Figure 17 : Middle leg cut and drilled.

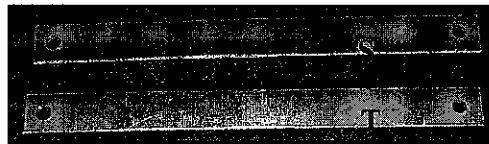


Figure 18 : Mechanical linkages.

Figure 18 shows the scrap PCB pieces cut and drilled to be used as the mechanical linkages. The following step will be assembling the legs. Upper pieces H and I are fasten to servo horns and also lower leg L and M to form assembled front legs. Upper pieces J and K are fasten to lower pieces N and O to form assembled back legs. Both front and back legs are then attached together using the mechanical linkages fasten using screws and nuts as shown in **Figure 19**. Middle leg piece, R is also fasten to a servo horn and 2 lower leg pieces P and Q to form an assembled middle leg as shown in **Figure 20**.

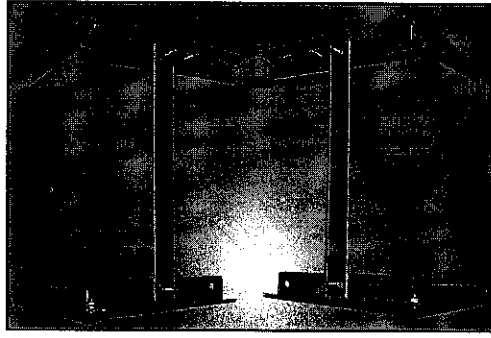
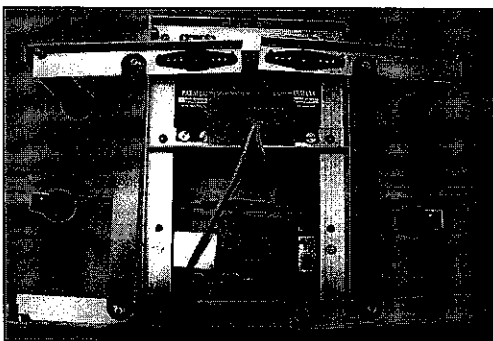


Figure 19 : Assembled left and right leg.

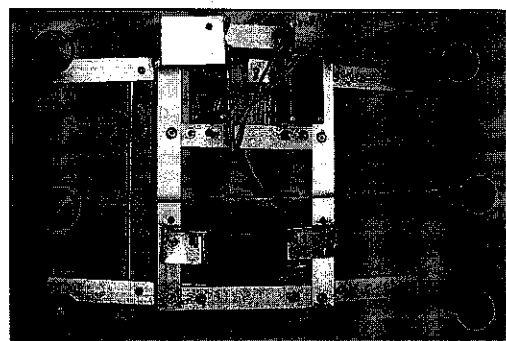


Figure 20 : Assembled middle leg.

The next step is to attach the legs to the servos on the chassis. Starting with the middle leg, servo mount is lined up with the middle servo and secured using the mounting screw. Then the front left leg assembly made up of pieces I and M is secured to the front left servo while the right leg assembly made up of pieces H and L is secured to the front right servo using mounting screws. The both the back legs are attached to piece C where the holes are drilled. The last step is to attach the rubber feet to the end of all lower legs. **Figure 21** shows the complete assembled structure.



Top View



Back View

Figure 21 : Complete assembled structure.

The materials chosen are to be light weighted but durable. Another criterion to be paid attention to is the foot of the structure. It must be able to sustain the structure from slipping and provide grip that can enable the structure to move forward even on slippery ground like tiles. Thus, the all the legs are attached with rubber feet to create better friction and grip during locomotion. All the materials are cut accurately to the measurement decided earlier. **Table 4** shows the measurement for all parts in the structure.

Table 4 : Measurement of structure parts.

Parts	Measurements
A, B, C	94 mm x 12 mm
D, E	150 mm x 12 mm
F	32 mm x 20 mm
G	20 mm x 20 mm
H, I	100 mm x12 mm
J, K	88 mm x 12 mm
L, M, N, O	88 mm x 9 mm
P, Q	44 mm x 9 mm
R	208 mm x 12 mm
S, T	131 mm x 12 mm

The mechanical design is the most crucial part of this project. It is very important to have a stable and robust structure before any other designs are added to develop the walking hexapod. All joints in the structure attached using screws are replaced by rivets. Rivets can hold tightly two pieces of aluminum together whereas screws tend to loosen up due to movements and vibration. Besides that, using rivets will reduce

the weight of the structure by eliminating screws, washers and nuts. This is indeed important as controls and power usage will be more efficient with lighter structure.

In order for the structure to be named as robot, it must have some intelligence. Sticking to the objective to achieve simple control and cost effective design, limit switches are used to install object avoidance capability into the hexapod. Two limit switches are used to be attached to the bottom of the two front servos as shown in **Figure 22**. Both are attached with the lever extended and pointed to opposite direction, 45 degrees to the left and 45 degrees to the right. Both the extended lever will act as tentacles of the structure. Whenever it touches something in front, it will be able to send signal to the controller to determine the action to be taken. Final mechanical design will have a structure look as shown in **Figure 23**.

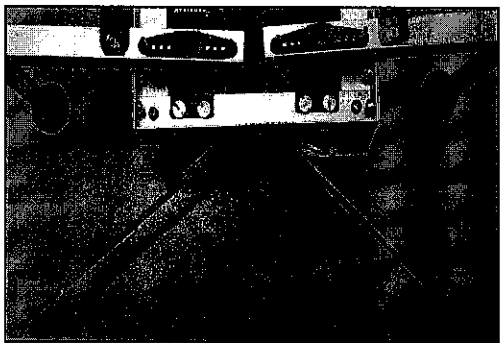


Figure 22 : Limit switches attached to servo motors.

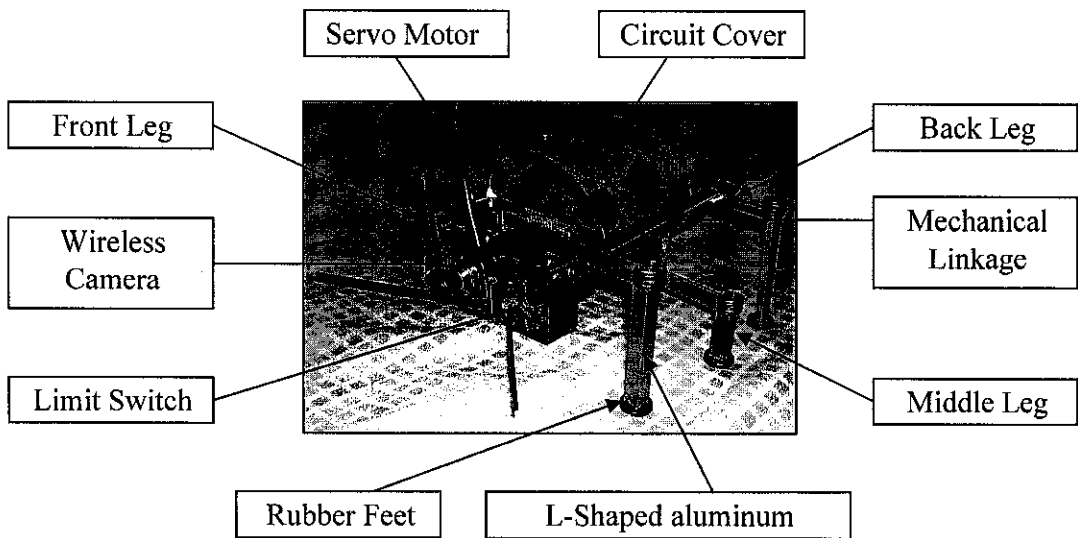


Figure 23 : Final mechanical structure.

4.2 Circuit Design

All the robot functions are controlled by Microchip 16F877 microcontroller. This microcontroller is an entire computer on a chip which eliminates large amount of hardware that might be required for the same functions. Microcontroller serves as the brain, controlling and managing all functions and sensors. The PIC16F877 used is clocked at 4 MHz and operates at 5V DC supply. The main controller board as shown in **Figure 24** is designed to support a keypad, an LCD display, and a few I/O ports as well as on board programming capability through the female pin connector.

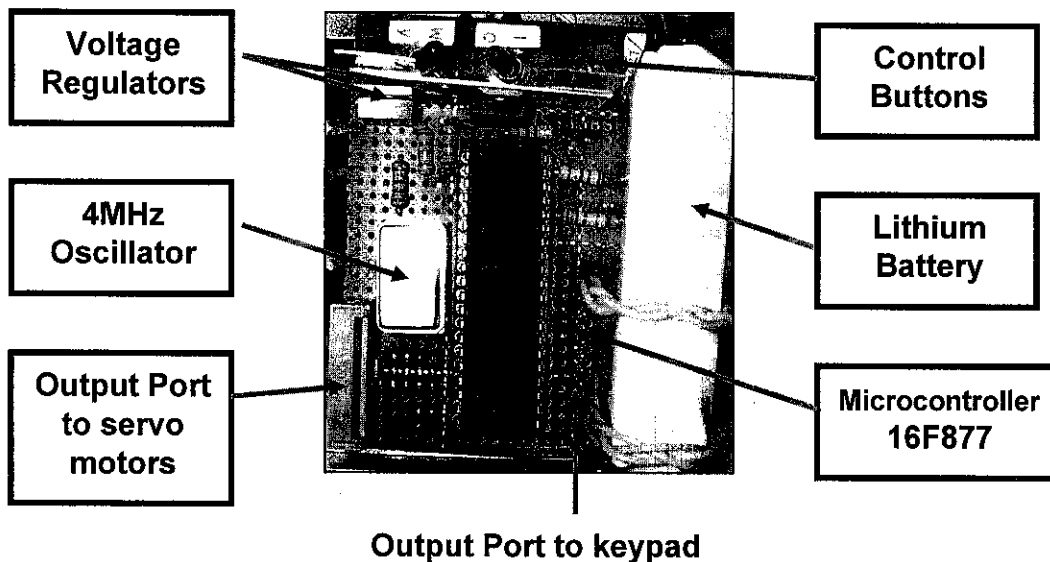


Figure 24 : Main controller board.

The PIC is chosen as the desired microcontroller in this project because of various factors. The advantage of designing around a microcontroller is that a large amount of electronics needed for certain applications can be eliminated. PIC 16F877 can be reprogrammed easily to perform different functions and is very inexpensive. It has a very small and limited instruction set. Even though the instruction set is small, it is very fundamental, which allows it to execute complex tasks efficiently. Secondly, the PIC is widely available, because it is a simple chip, easy to use, with high demand all over the world.

Referring to the schematic shown in **Figure 25**, five I/O ports are available to be used as inputs and outputs. Port A is connected to two limit switches for the object avoidance application. Port C will be used as output ports connecting to servo motor to transmit pulses to control them. As for Port D, it is used as input ports connected to

keypad. The board is fully utilized to control the movement of the robot either manually by human or automatically using the object avoidance ability. There are two voltage regulator used. LM7805 is used to convert 12V input to 5V output to be supplied to the controller and also servo motors while LM7809 is used to convert 12V input to 9V output to be supplied to the wireless camera used for remote surveillance. For clearer picture, please refer to **Appendix E**.

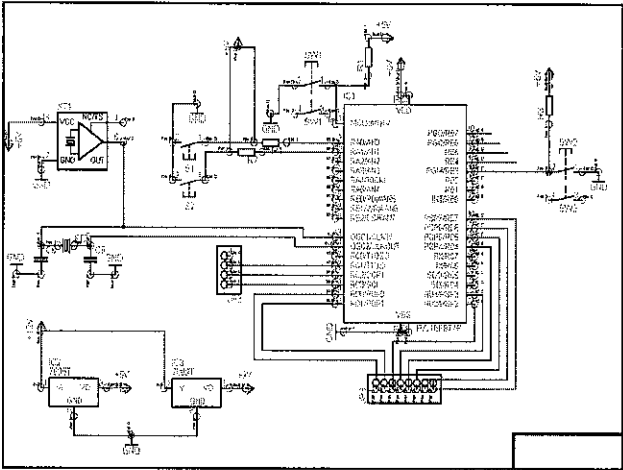


Figure 25 : Schematic for main controller board.

4.3 R/C Servo Motor

Standard servo motor used in the design is PARALLAX standard servo motors. For servo motors specification, please refer to **Appendix B**. A typical servo has three wires are V+, Control, and Ground. This servo typically runs on 4.8V. The control line is used to position the servo and normally attached to the processor which is the microcontroller. Servo motors are controlled by sending them a ‘pulse’ of variable width. The parameters for pulse are that it has a minimum width, a maximum width and a repetition rate. The convention is that a pulse of approximately 1.5ms is the neutral point for the servo.

When the pulse sent to a servo is less than 1.5ms, the servo positions and holds its output shaft some numbers of degrees counter clockwise from the neutral point. When the pulse is wider than 1.5ms, the opposite occurs. Generally, the minimal pulse will be about 1ms and maximal will be 2ms. The control wire is used to communicate the angle and the angle is determined by the duration of a pulse that is applied. A servo motor expects to see a pulse every 20 milliseconds.

Thus, to control the servo motors' angle of rotation, the PWM has to be generated from the microcontroller repetitively every 20 milliseconds. In this project, there are only three different pulses needed to be generated. As known, 1500 μ s wide pulse will position the servo at neutral, 90 degrees. To rotate the shaft to the left, 1400 μ s wide pulse is applied while to rotate the shaft to the right, 1600 μ s wide pulse is applied. **Figure 26** shows the duration of pulses generated by microcontroller after being programmed and captured using oscilloscope.

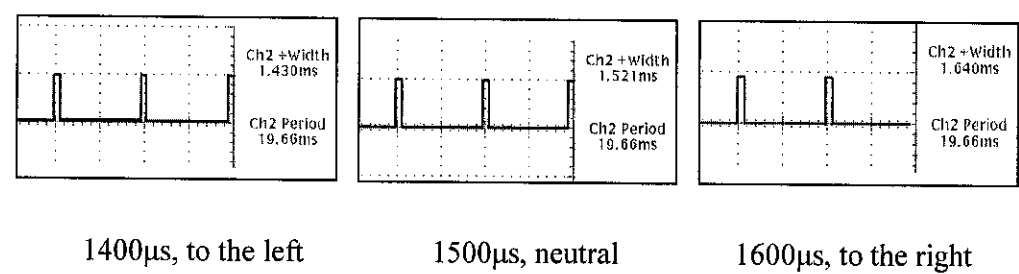


Figure 26 : Duration of PWM generated to control servo motor.

4.4 Walking Algorithm Design

Robot's walking gait is the sequence of leg movements it uses to move from one place to another. Each leg movement is broken into steps cycles, where a cycle is complete when the leg configuration is in the same position as it was when the cycle was initiated. A walking gait is repetitive pattern of foot placement causing a regular progression forwards or backwards. With six legged, the most popular gait used will be the alternating tripod gait. Six legs allow robot to have three legs on the ground at all times, making it a stable 'tripod'.

With the alternating tripod gait, the legs are divided into two sets of three legs. Each set is comprised of the front and rear legs on the side, along with the middle leg on the opposite side. This is the fastest stable gait. During the step, the weight of the robot is only supported on three legs. If one fails to find firm footing, the robot will fall. The middle legs lift the body and are used as a swivel, but they never actually move in a forward or reverse direction.

The primary leg positions and the corresponding PWM needed for programming the walking movements are chosen to be 1400 μ s, 1500 μ s and 1600 μ s. These pulse-width values can be used to program the leg movements.

4.4.1 Forward and Reverse Walking Sequence

To start the forward walking sequence, all legs are flat on the ground with the servos in their middle positions as shown in Frame 1 in **Figure 27**. The first move shown in Frame 2 is to move leg 2 down, which lifts legs 1, 3 and 5 up off the ground. In Frame 3, leg 2 remains down and is used as a swivel as legs 4 and 6 are moved backwards, propelling the right side of the robot forward. In Frame 4, leg 2 remains down while legs 1 and 3 are moved forward in anticipation of the next move. In Frame 5, leg 5 is now moved down, lifting 2, 4 and 6 up off the ground. In Frame 6 leg 5 remains down and legs 4 and 6 are moved forward in anticipation of the next move. In Frame 7, leg 5 remains down and acts as a swivel as legs 1 and 3 are moved backwards, propelling the left side of the robot forward. For the robot to continue walking forward, the sequence is repeated from Frame 2 to Frame 7.

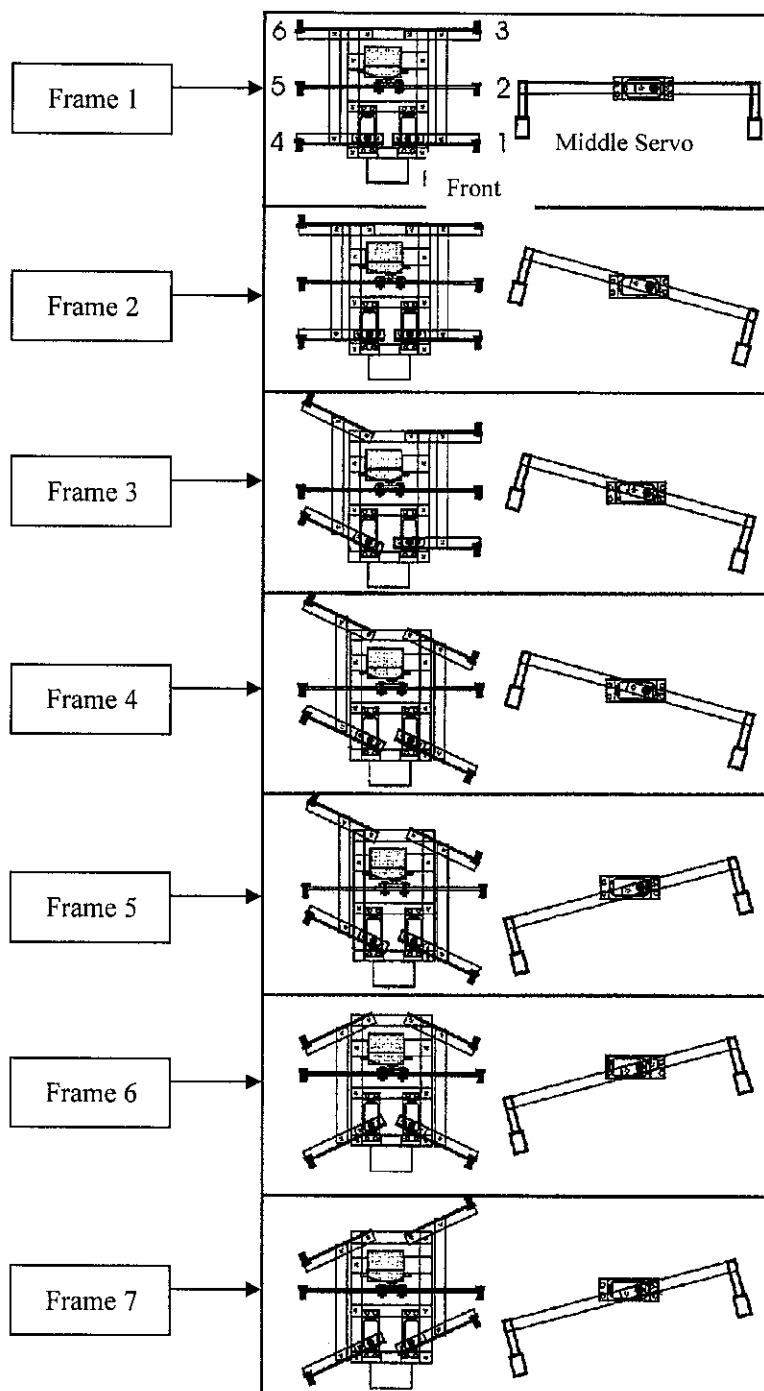


Figure 27 : Leg position sequence to achieve forward walking.

Software has to set the servo position six times to complete one forward walking sequence. The pulse-width values (PWM) for each frame of servo position setting for forward walking sequence are listed in **Table 5**. Pulse-width generated is captured using oscilloscope and the signals can be observed in **Figure 28**.

Table 5 : Forward walking sequence with pulse-width value.

Frame	Middle Servo (μ s)	Left Servo (μ s)	Right Servo (μ s)
1	1600	-	-
2	1600	-	1600
3	1600	1600	-
4	1400	-	-
5	1400	-	1400
6	1400	1400	-

For robot to walk in reverse, the sequence in Figure 25 is run in reverse for the left and right servo legs while the middle servo PWN values remain the same. **Table 6** shows the pulse-width values for each servo positioning for reverse walking sequence.

Table 6 : Reverse walking sequence with pulse-width value.

Frame	Middle Servo (μ s)	Left Servo (μ s)	Right Servo (μ s)
1	1600	-	-
2	1600	-	1400
3	1600	1400	-
4	1400	-	-
5	1400	-	1600
6	1400	1600	-

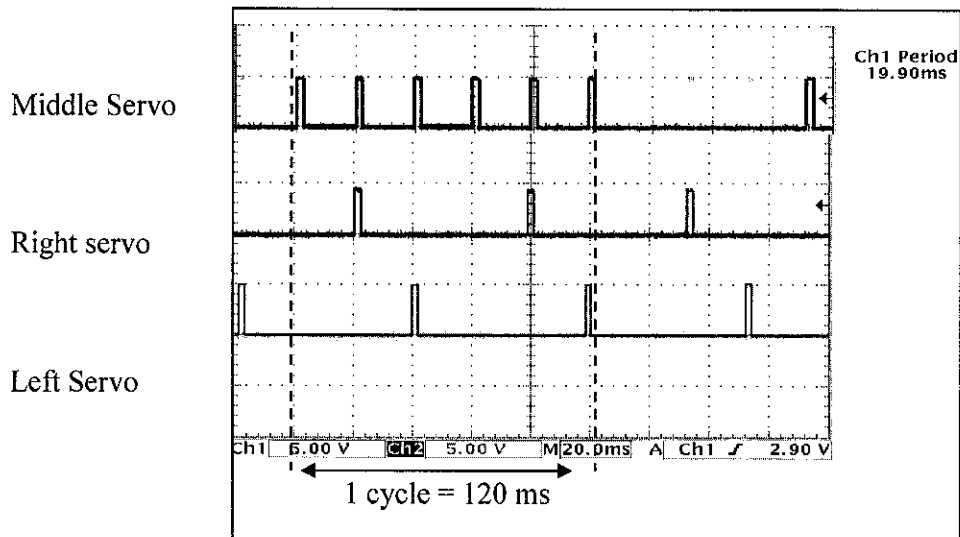


Figure 28 : PWM signals generated for forward walking sequence.

4.4.2 Rotating Left and Right Walking Sequence

Assuming the turning motion will be carried out from the starting position where all the robot's legs are flat on the ground. Referring to **Figure 29**, the robot will move leg 2 down, lifts legs 1, 3 and 5 off the ground. In Frame 2, leg 2 remains in the down position, acting as a pivot point while legs 4 and 6 are moved backwards, swiveling the robot to the left. At the same time, legs 1 and 3 are moved backwards in anticipation of the next move. In Frame 4, the middle servo moves leg 5 to the downward position, lifting legs 2, 4 and 6 off the ground. Then, while leg 5 remain downward and leg 1 and 3 move forward, the robot will pivot to the left. At the same time, legs 4 and 6 are moved forward in anticipation of the next move. For robot to turn to the right, the sequence is reversed for the left and right servos, while middle servo values stay the same.

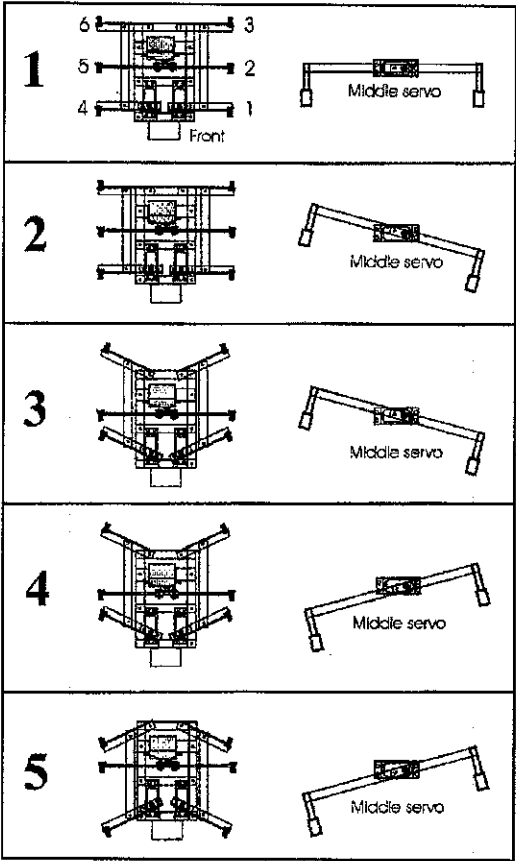


Figure 29 : Robot leg positions for turning sequence.

Table 7 and **Table 8** show the exact four step sequence and the corresponding pulse-width values that are used to control the turning locomotion.

Table 7 : Pulse-width values for left-turn walking sequence.

Frames	Middle Servo(μ s)	Left Servo(μ s)	Right Servo(μ s)
1	1600	-	-
2	1600	1400	1600
3	1400	-	-
4	1400	1600	1400

Table 8 : Pulse-width values for right-turn walking sequence.

Sequence Number	Middle Servo(μ s)	Left Servo(μ s)	Right Servo(μ s)
1	1600	-	-
2	1600	1600	1400
3	1400	-	-
4	1400	1400	1600

Pulse-width generated is captured using oscilloscope and the signals can be observed in **Figure 30**.

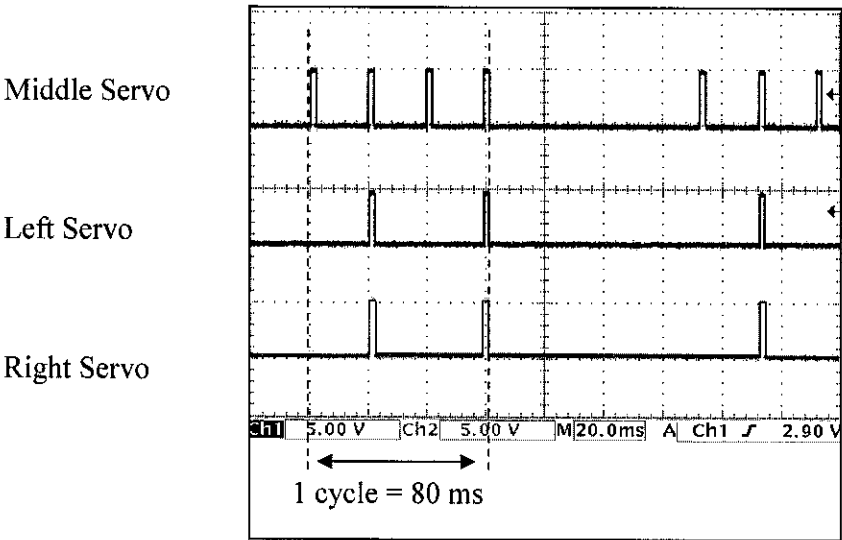


Figure 30 : PWM signals generated for turn left walking sequence.

4.5 PIC Programming

Microchip technology has developed a line of reduced instruction set computer (RISC) microprocessors called the programmable interface controller (PIC) [13]. As stated earlier, PIC16F877 will be used as the microcontroller for the design control. Please refer to **Appendix D** PIC16F877 datasheet. This device can be reprogrammed over and over again as it uses flash read only memory for program storage. Thus, this makes it ideal for experimenting as the chip does not need to be erased using

ultraviolet light source. PIC16F877 is a 40-pin device with a 16-bit data bus and registers. The device is clocked using the 4MHz oscillator. The PIC is equipped with five different ports. Port A has 5 I/O lines, port B has 8 I/O lines, port C has 8 I/O lines, port D has 8 I/O lines and port E has 3 I/O lines. Programming is done using C language programming. C language is a high-level language used to provide amazingly greater amounts of functionality and speed.

This device is called the pulse stream PIC, because it is responsible for generating the pulse streams to all 3 servo motors simultaneously. The type of pulse stream to each servo motor depends on key press controlled by human or input from limit switches. This hexapod is programmed to be functioned in two different modes which are the manual control and the auto object avoidance mode. Flow charts have been developed before the programming is done. Three different loops are created to control the movement of hexapod. **Figure 31** shows the flow charts created to control the MAIN loop in the program.

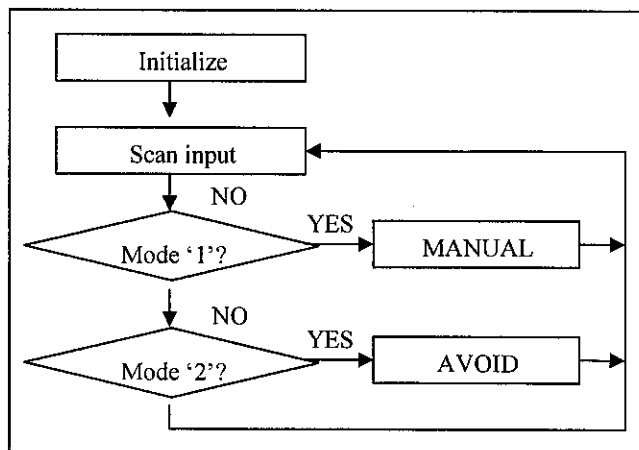


Figure 31 : Flow diagram of MAIN Loop.

The MAIN loop will be executed once the main controller board is turn ON. If the board is set to Mode 1, the program will jump to the MANUAL loop and if set to Mode 2, AVOID loop, which puts the hexapod on the automatic mode will be accessed. The MANUAL loop is used to control movements according to input by human using keypad. Port D is used to connect to the keypad. By sensing the key press button, particular movement can be executed. **Figure 32** shows the flow diagram created for MANUAL loop programming. Please refer to **Appendix F** for the MAIN loop C codes.

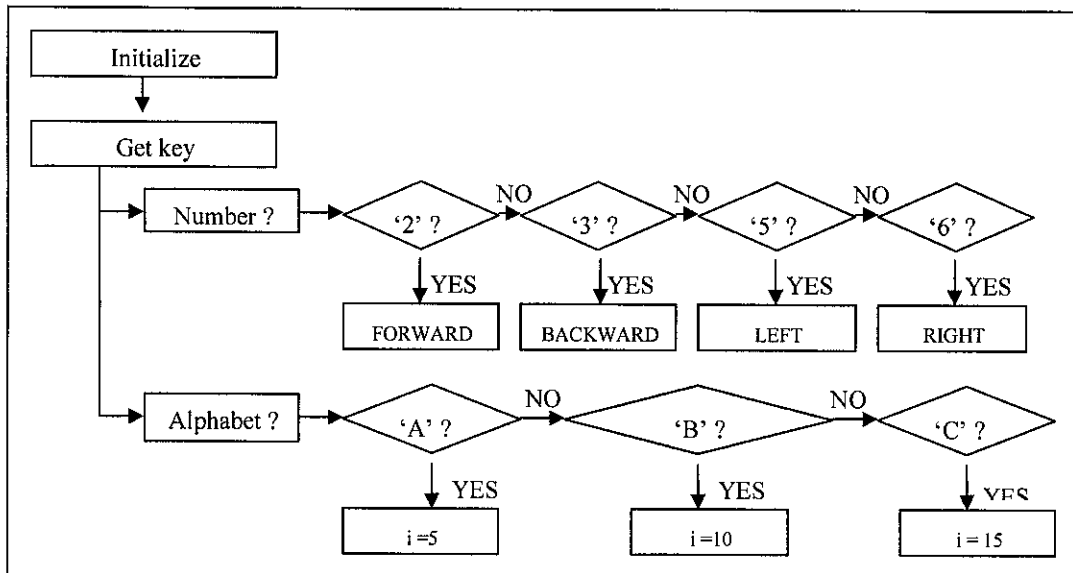


Figure 32 : Flow diagram for MANUAL loop.

Program will keep scanning the status of the keypad. If key '2' is pressed, codes to create the forward movement will be accessed. Thus, the signals will be output to the 3 servo motors and the structure will be moved forward according to the algorithm designed earlier. Same concept will be applied to the rest of the movements, key press '3' will execute backward motion, key press '5' will execute the left turning motion and key press '6' will execute the right turning motion. However, if it was the alphabet that is pressed, the speed of the locomotion will change accordingly. Key press 'A' is for the highest speed, key press 'B' for the medium speed and key press 'C' is the low speed control. Please refer to **Appendix G** for MANUAL loop C codes.

Program 1 show the C codes written to generate the forward walking motion. Please refer to **Appendix H** for C codes written to generate other walking motion.

Program 1 : forward () program lists

```

for(a=0;a<i;a++)
{
    output_high (PIN_C1);
    delay_us(1600);
    output_low (PIN_C1);
    delay_ms (18);
}
// i = number of loops to hold the
// servo position (speed control)
// pulse-width 1600us is applied to
// middle servo
// delay before sending another PWM

```

```

for(a=0;a<i;a++)
{
    output_high (PIN_C1);          // codes for frame 2 positioning
    output_high (PIN_C3);          // 1600us for middle and right servo
    delay_us(1600);
    output_low (PIN_C1);
    output_low (PIN_C3);
    delay_ms (18);
}
for(a=0;a<i;a++)
{
    output_high (PIN_C1);          // codes for frame 3 positioning
    output_high (PIN_C2);          // 1600us pulse-width applied to
    delay_us(1600);                // middle and left servo
    output_low (PIN_C1);
    output_low (PIN_C2);
    delay_ms (18);
}
for(a=0;a<i;a++)
{
    output_high (PIN_C1);          //codes for frame 4 positioning
    delay_us (1400);               // 1400us pulse-width applied to
    output_low (PIN_C1);           // middle servo
    delay_ms (18);
}
for(a=0;a<i;a++)
{
    output_high (PIN_C1);          //codes for frame 5 positioning
    output_high (PIN_C3);          // 1400us pulse-width applied to
    delay_us(1400);               // middle and right servo
    output_low (PIN_C1);
    output_low (PIN_C3);
    delay_ms (18);
}
for(a=0;a<i;a++)
{
    output_high (PIN_C1);          //codes for frame 6 positioning
    output_high (PIN_C2);          // 1400us pulse-width applied to
    delay_us(1400);               // middle and left servo
    output_low (PIN_C1);
    output_low (PIN_C2);
    delay_ms (18);
}
}
*****

```

Note that all pulse generating codes are placed in loops. The loops are meant to hold the position of the shaft for a certain amount of time to position the structure. Single PWM is unable to position the structure stably as it is too short to place the structure in the desired algorithm pattern. Thus, the minimum loops required to enable the structure to position itself stably for each movement is at least 5 loops. The default speed is set to medium; 10 loops, if it's not controlled by human. **Figure 33** shows the signals captured using oscilloscope during forward locomotion. **Figure 34** shows the signals captured for turning left motion.

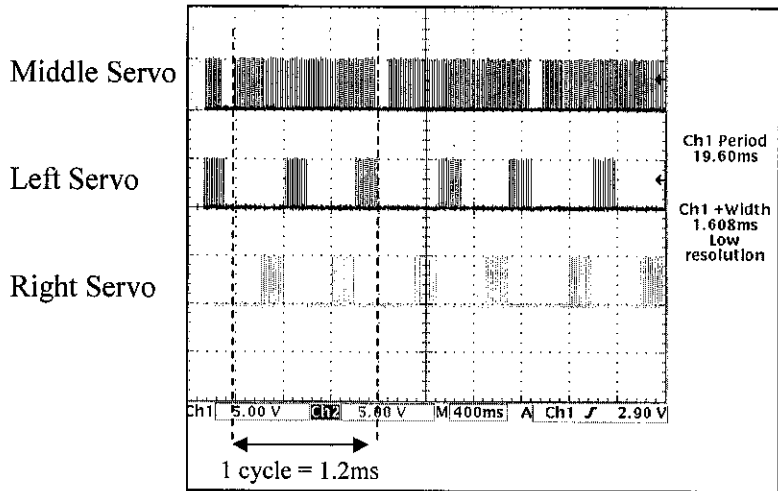


Figure 33 : PWM generated to servo motor during forward locomotion.

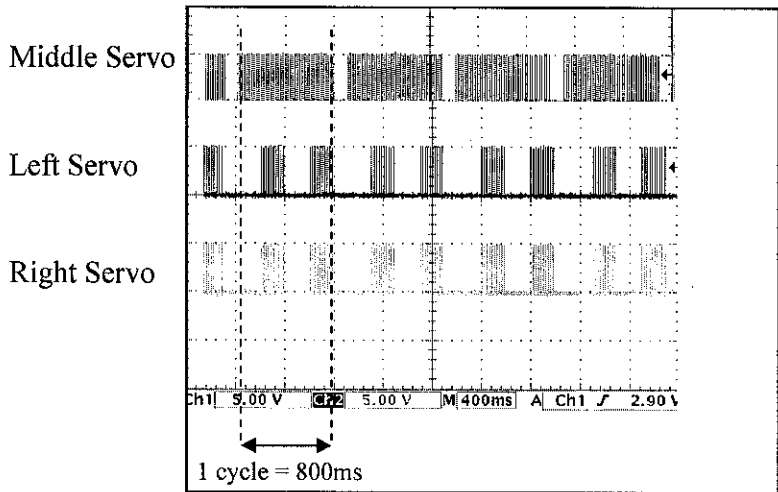


Figure 34 : PWM generated to servo motor during left turning locomotion.

If the main controller board is set to Mode 2, the automatic mode is selected, the AVOID loop is accessed. When this loop is executed, the inputs from users will be ignored. Basically the structure will keep moving forward unless it encounters any object in front of it which can be sensed by the two tentacles attached at the front of the structure. The tentacles are actually the extension of the lever of the limit switches used to sense the presents of object. **Figure 35** shows the flow diagram of AVOID loop.

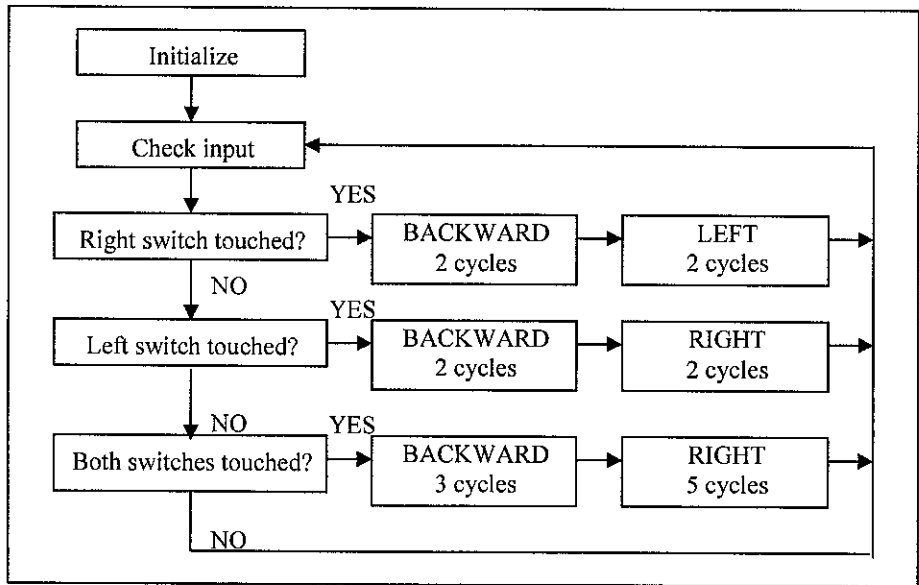


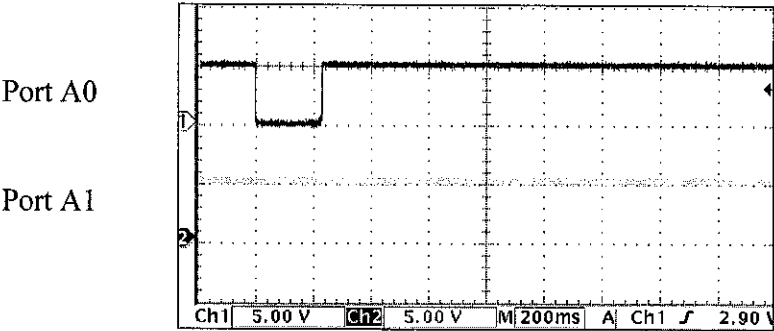
Figure 35 : Flow diagram of AVOID loop.

The AVOID loop will be accessed as long as there is no interruption from human. Both the limit switches are connected to Port A. The right limit switch is connected to Port A0 while the left switch is connected to Port A1. Both these ports are connected to +5V, which means having the logic ‘1’ all the time as shown in the schematic in **Appendix E**. The program will run in loops to check the status of the limit switches. If the right switch or the right tentacle is touched, Port A0 will be grounded; logic ‘0’ will be sensed. Hexapod will locate the object is on the right. If the left tentacle is touched, Port A1 will be grounded giving logic ‘0’, concluding the object is on the left side. However, if both switches are touched, giving both Port A0 and Port A1 logic ‘0’, hexapod will know the object is right in front of it. Refer to **Table 9** on how the structure locates the object based on the limit switches input.

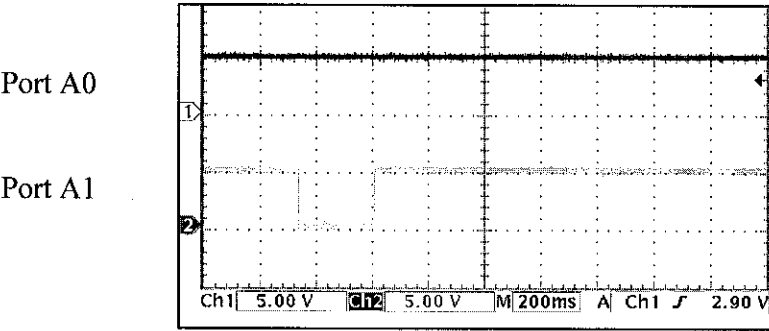
Table 9 : Object location based on the input from limit switches.

Tentacle touched		Input signals received		Hexapod Locating the Object
LEFT	RIGHT	PORT A1	PORT A0	
NO	YES	1	0	Object is on the right
YES	NO	0	1	Object is on the left
YES	YES	0	0	Object is right in front

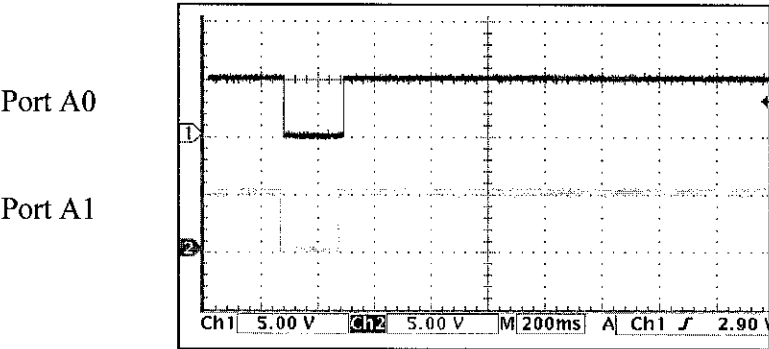
Figure 36 shows the signal of the limit switches input captured using oscilloscope. Note that the input signals are always high, giving logic '1' and whenever it touches object, the signal will be low giving logic '0'.



Right tentacle touched object, Port A0 logic 0.



Left tentacle touched object, Port A1 logic 0.



Both tentacles touched object, Port A1 logic 0.

Figure 36 : Input signals from limit switches during AUTO mode.

Based on the interpretation made by the structure, the hexapod will be able to behave and make the determined action to find its new path or avoid the object and continue to walk forward. **Table 10** shows the robot behavior based on the object location interpreted.

Table 10 : Robot behavior based on the object location.

Object Location	Robot Behavior
RIGHT	Reverse 2 steps, turn to the left, and continue walking forward.
LEFT	Reverse 2 steps, turn to the right, and continue walking forward.
DIRECTLY IN FRONT	Reverse 5 steps, turn 90 degrees to the right, and continue walking forward.

Please refer to **Appendix I** for C codes written to control the object avoidance ability or the AUTO mode named AVOID loop. **Appendix J** shows the C codes used to display characters on the LCD screen and **Appendix K** shows the C codes used to retrieve key press from human control.

4.5.1 Control Optimization

To simplify the sequence for the purposes of programming and to speed up the walking gait, the steps in which legs are moved forward in anticipation of the next move can be performed at the same time that the opposite set of legs is propelling the robot forward. Thus, the software generated will only have to set the servo positions four times to complete one forward walking sequence instead of six times. For robot to walk in reverse, the forward sequence is run in reverse for the left and right servo legs while the middle servo values remain the same.

For easier understanding, referring to **Figure 27** in **Section 4.4.1**, Frame 3 and Frame 4 is combined into one and Frame 6 and Frame 7 are combined into another chunk. Thus, the whole cycle of consists of only 4 frames as shown in **Figure 37**.

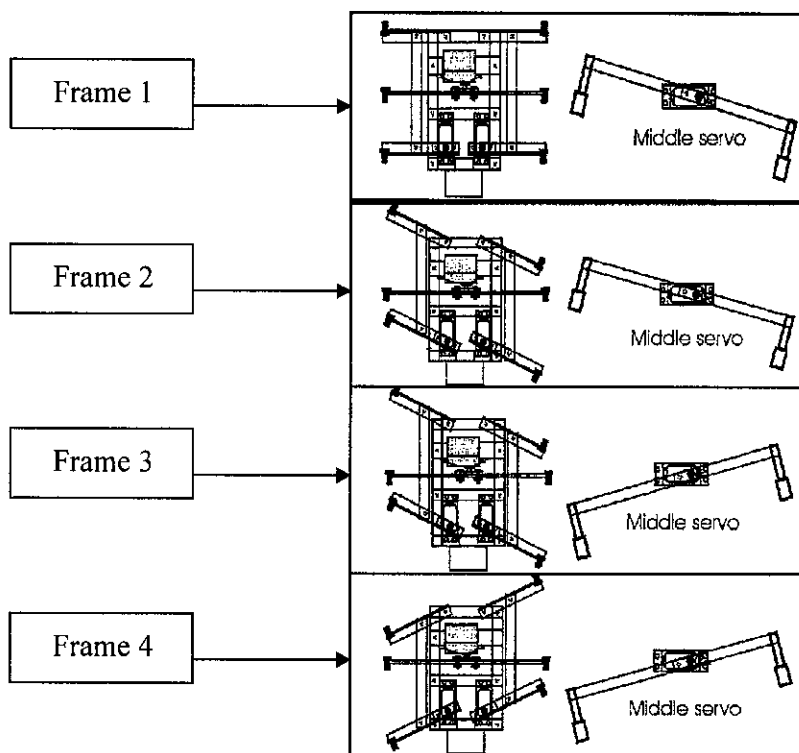


Figure 37 : Simplified sequence to achieve forward walking.

Referring to **Section 4.4.1**, the forward and reverse walking gait consists of 6 frames to achieve 1 cycle of movement. This will indeed need six different chunks of program to control the servo motors according to the particular frame. Each position needs 10 loops of 20ms to position the servo motors. Thus, the whole cycle will require 1.2 seconds to complete. By combining the sequences, the software will require only 4 frames to complete one cycle of forward/reverse movement. This will results in faster walking and shorten the process time to only 800ms. However, for the turning left and turning right, the period of execution for one cycle remains the same. **Table 11** and **Table 12** below shows the pulse-width value supplied to the servo motors for forward and reverse motion.

Table 11 : Pulse-width values for forward walking sequence.

Frame	Middle Servo(μ s)	Left Servo(μ s)	Right Servo(μ s)
1	1600	-	-
2	1600	1600	1600
3	1400	-	-
4	1400	1400	1400

Table 12 : Pulse-width values for reverse walking sequence.

Frame	Middle Servo(μ s)	Left Servo(μ s)	Right Servo(μ s)
1	1600	-	-
2	1600	1400	1400
3	1400	-	-
4	1400	1600	1600

Figure 38 shows the PWM generated to control the servo motors for forward/reverse walking motion using the 4 frames per cycle sequence.

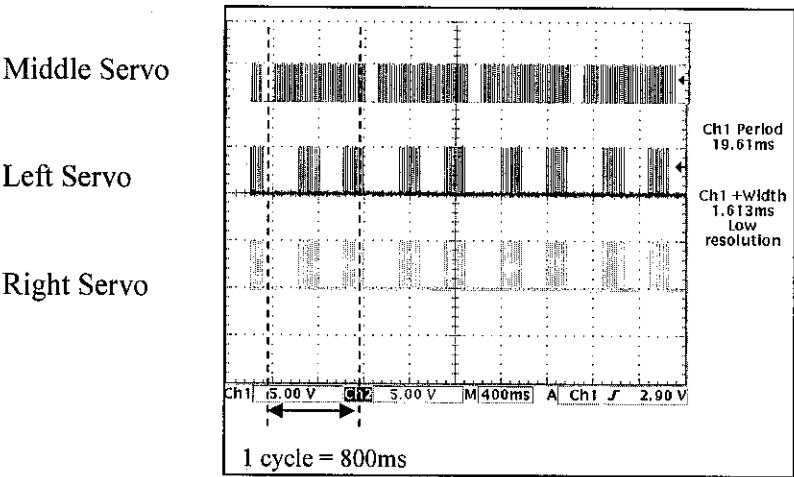


Figure 38 : PWM generated for forward/reverse motion using 4 frames/cycle.

From the graphs above, it can be seen that the improvement made by combining the frames reduce the execution time per cycle by 400ms. This means that for every 2 cycles execution of 6 frames algorithm; duration of 1200ms x 2, the program can execute 3 cycles of 4 frames algorithm; duration of 800ms x 3. This can be notice on the hexapod movement as the positioning is more aggressive, effective and faster to reach destination point. Thus, for the rest of the programs written are based on the 4 frames per cycle algorithm to achieve the best and most efficient walking gait. The improvement stated is not applied to turning left and right algorithm because it has only 4 frames and it is at its best performance.

4.5.2 Speed Control Analysis

One of the features in the design is the speed control. Three speed is available, the low speed, medium speed and high speed. Microcontroller defines low speed setting by increasing the number of looping needed for each positioning or rather each frame. As stated earlier, each cycle will consist of 4 frames of position. Each position will need 10 loops of 20ms to control the servo motor and this is the default speed. The loops are meant to hold the position of the shaft for a certain amount of time to position the structure. If the speed is set to low, microcontroller will then output signals in 15 loops to hold each position. If the speed is set to high, microcontroller will then output signals in 5 loops for each position which is shorter time. **Figure 39** shows signals generated for HIGH speed forward movement.

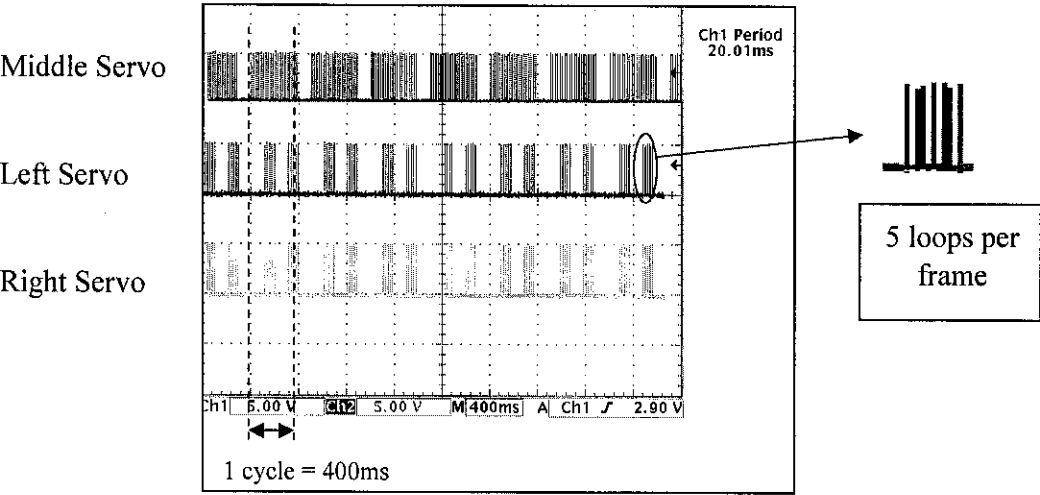


Figure 39 : Signals generated for HIGH speed forward/reverse locomotion.

From figure above, notice that the signals for each position only has 5 stems. Thus, the total period of time needed for 1 cycle will be $5 \times 20\text{ms} \times 4 \text{ frames} = 400\text{ms}$. As for signals generated for turning left and turning right motion, the duration and looping applied will be the same as how it is applied for forward/reverse algorithm for low, medium and high speed.

Figure 40 shows PWM generated to servo motor for medium speed control for forward/reverse locomotion. Notice that the signals for each position have 10 stems. Thus, the total period of time needed for 1 cycle will be $10 \times 20\text{ms} \times 4 \text{ frames} = 800\text{ms}$.

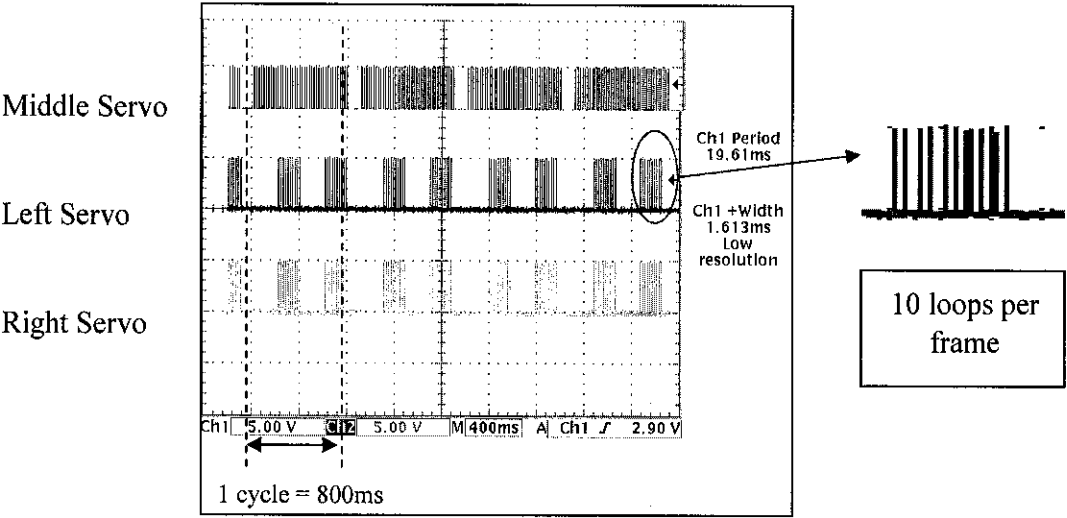


Figure 40 : Signals generated for MEDIUM speed forward/reverse locomotion.

Figure 41 shows PWM generated to servo motor for low speed control for forward/reverse locomotion. Notice that the signals for each position have 15 stems. Thus, the total period of time needed for 1 cycle will be $15 \times 20\text{ms} \times 4 \text{ frames} = 1200\text{ms}$.

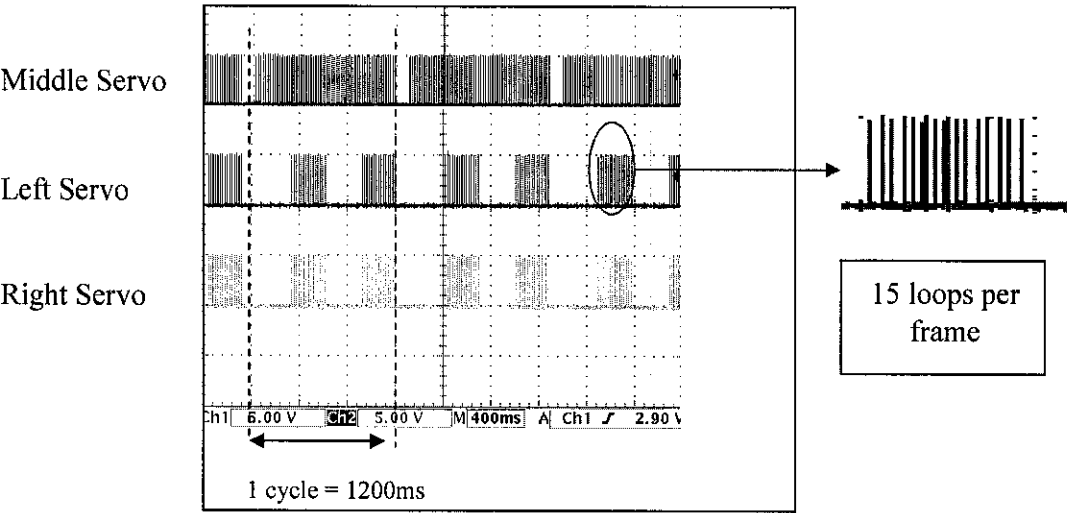


Figure 41 : Signals generated for LOW speed forward/reverse locomotion.

4.6 Shortcomings, Challenges and Obstacles

The original design of the structure is a little different from the final one. Supposing the mechanical linkage is connected using screws as it is a moving part. However, the movement of forward and backward of the legs attached to the linkage often loosens the screws. This causes the movement to be shaky and the back leg not being able to position correctly. Thus, in order to solve this problem, the mechanical linkage is attached to the legs using rivet. Since connection using rivet is very tight and strong, the holes drilled for the linkage is widen to provide more space. After riveting, the linkage is move vigorously to loosen up so that both the pieces connected to the legs can move horizontally and not vertically.

Another problem faced in designing is the power control. As the 5V DC supply is connected to the main controller board and the 3 servo motors, the supply is unable to withstand continuous powered servo motor for too long. When that happens, the structure tends to vibrate vigorously as the servo motor is losing power to hold the position and the weight of the structure. Thus, the programming is modified to create pulses with loops instead of supplying continuous pulses to 3 servo motors all the time. This effect can be seen when hexapod is operating at different speed.

As notice, when lower speed is applied, the movement of the structure is not that efficient because more current will be drawn to hold the servo position. The most efficient speed is the medium speed as robot's movement is the smoothest among the 3 speed. When high speed is applied, the movements tend to be more aggressive and the impact of foot hitting the ground increases. These can be further improved by using more servo motors to adjust the movement but that will increase costs and more complex controls.

CHAPTER 5

CONCLUSION

Final Year Project (FYP) is designed to provide an opportunity to the students to design and handle a project just like the working world. With the knowledge that students have learnt throughout their studies in UTP, they can apply it practically in their projects. Specifically for this project, designing a hexapod, it involves all aspects from mechanical design up to programming. Hexapod robot project requires some fundamental of mechanical design, applying practical usage of motors and sensors, electronic circuit design as well as microcontroller programming. This project not only provides the opportunity to apply the knowledge practically but also enhance the problem solving and designing skills.

The most important part of this development of walking machines is the mechanical design. The structure has to be light in weight, rigid and robust and also simple and in great balance. This includes critical decision in choosing the right material, and using the correct balancing concept. The material chosen is aluminum as it is light in weight, strong and rigid as well as available in nearby hardware store. All sawing and drilling is in the workshop in Building 22, UTP.

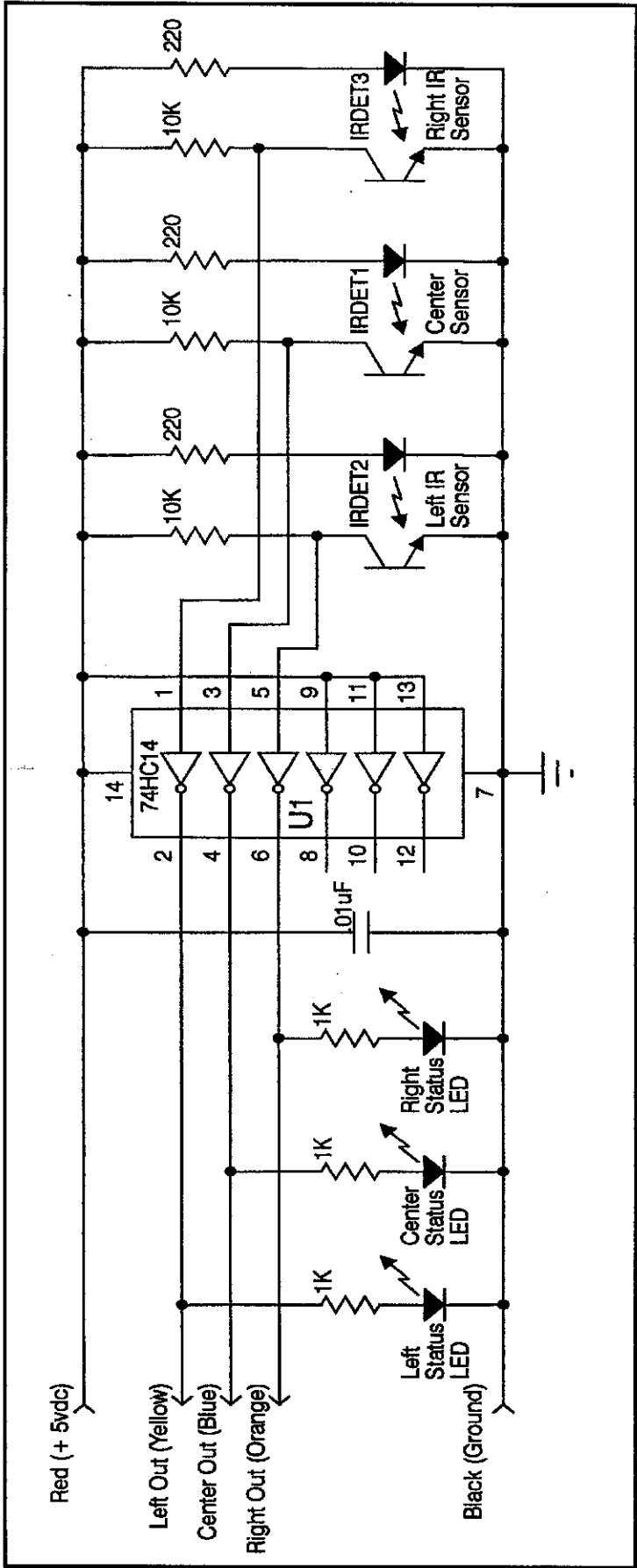
Tripod technology is implemented in designing the algorithm for its locomotion to achieve a stable and simple walking gait. Moreover, the walking gait is designed using the appropriate speed to control so that the movements are at its most efficient and applicable condition. This structure is programmed to walk forward, reverse and also turning left and right. It also has the ability to walk in three different speeds controlled by human or walking in autonomous mode with object avoidance ability. With the wireless camera attached on the structure, it is able to be used for remote monitoring and surveillance as well. The objective of the project is fully achieved successfully where the finished product is cost effective, light, robust, has easy control and intelligently applicable.

REFERENCES

- [1] <http://www.cis.plym.ac.uk/cis/InsectRobotics/Background.htm>
- [2] Karl Williams, *Insectronics – Build Your Own Walking Robot*, Mc Graw Hill, 2003, page 4.
- [3] Mark E. Rosheim, *Robot Evolution The Development of Anthrobotics*
- [4] <http://www.faculty.ucr.edu/~currie/roboadam.htm>
- [5] <http://www.thetech.org/robotics/universal/page02.html>, universal robots: the history and working of robotics, page 2.
- [6] <http://www.robotgames.net/robotgames> , Dr. Mark Tilden, Walker Triathalon 2002 Rules and Resources, page 28, Section: Background.
- [7] Roger D. Quinn, Gabriel M. Nelson, Richard J. Bachmann, Daniel A. Kingsley, John Offi, and Roy E. Ritzmann, *Insect Designs for Improved Robot Mobility*, Berns and Dillmann eds., Prof. Eng. Pub., 69-76, 2001.
- [8] Karl Lunt, 2000, *Build Your Own Robot*, A K Peters Natick, Massachusetts
- [9] Éric Lespérance, Alexis Lussier Desbiens, Marc-André Roux, Marc-André Lavoie and Philippe Fauteux, *Design of a Small and Low-Cost Power Management Unit for a Cockroach-Like Running Robot*, IROS 2005
- [10] D. L. Jindrich and R. J. Full, “Many-legged maneuverability: dynamics of turning in hexapods,” *Journal of experimental biology*, no. 202, pp. 1603-1623, 1999.
- [11] Martin Buehler, *Dynamic Locomotion with One, Four and Six-Legged Robots*, McGill University Montreal
- [12] <http://www.ranchbots.com/walker/walker.htm>, Robot Projects, 6 Legged Walkers using 3 Servo Motors, Section: Theory, page 1
- [13] Richard Barnett, Larry O’Cull and Sarah Cox, *Embedded C Programming and the Microchip PIC*, Delmar Learning, 2004.

APPENDICES

APPENDIX A LINE TRACKING CIRCUIT



APPENDIX B GANTT CHART

NO	Activities / Week	Semester I													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Preliminary Research Work														
	Introduction														
	Objective														
	List of references/literature														
3	Project planning														
	Submission of Progress Report I				+										
	Project Work														
	Reference/Literature														
4	Design and Fabrication of Biped Drives and Structure														
	Circuit Design and Software Familiarization														
	Project work continue														
	Testing of Robot Mobility and Movements														
6	Programming for basic movements														
	Add intelligence and Fine Tuning														
	Submission of Interim Report Final Draft											+			
	Submission of Interim Report												+		
9	Oral Presentation														

APPENDIX C

PARALLAX SERVO MOTOR



555 Marie Drive, Suite 150
Palo Alto, California 94304, USA
Office: (650) 434-0200
Fax: (650) 434-0000

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
International:
<http://www.parallax.com/intl.aspx#intl>

Standard Servo (#900-00005)

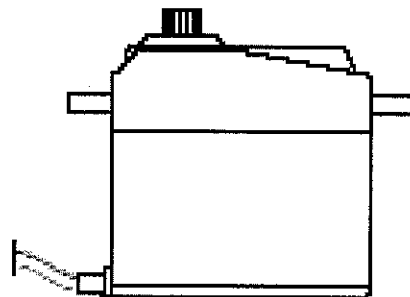
General Information

The Parallax standard servo is ideal for robotics and basic movement projects. These servos will allow a movement range of 0 to 180 degrees. The Parallax servo output gear shaft is a standard Futaba configuration. The servo is manufactured by Futaba specifically for Parallax.



Technical Specifications

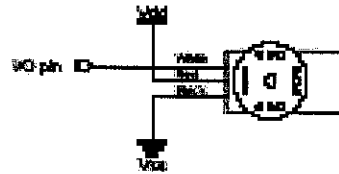
- > Power 6Vdc max
- > Speed 0 deg to 180 deg in 1.5 seconds on average
- > Weight 45.0 grams/1.59oz
- > Torque 3.40 kg-cm/17oz-in
- > Size mm (L x W x H)
40.5x20.0x38.0
- > Size in (L x W x H)
1.60x.79x1.50



Motor Control from a BASIC Stamp

Parallax (www.parallax.com) publishes many circuits and examples to control servos. Most of these examples are available for download from our web site. On www.parallaxinc.com type in "servo" and you'll find example codes below.

Wiring setup



The servo is controlled by pulsing of it's signal line. If you are using an Basic Stamp this is done with the `pulsout` command. Below is stamp code that will help you with basic control of a servo. The codes below may not move the servos from one extreme to another but it will give you a general demonstration on function.

Stamp1 code

```
SYSIO Servo pin = 0      'I/O pin that is connected to servo
SYSIO Temp = 40         'Work space for FOR NEXT
start:
  FOR temp = 70 TO 210
    PULSOUT Servo pin,temp
    PAUSE 50
  NEXT
  FOR temp = 250 TO 10
    PULSOUT Servo pin,temp
    PAUSE 50
  NEXT
  GOTO start
```

'Stamp 2,2a,2pe

```
Servo pin  COM      0      'I/O pin that is connected to servo
Temp      VAR      Word    'Work space for FOR NEXT
start:
  FOR temp = 200 TO 1300
    PULSOUT Servo pin,temp
    PAUSE 50
  NEXT
  FOR temp = 1300 TO 100
    PULSOUT Servo pin,temp
    PAUSE 50
  NEXT
  GOTO start
```

'Stamp 2xx,2p24/40

```
Servo pin  COM      0      'I/O pin that is connected to servo
Temp      VAR      Word    'Work space for FOR NEXT
start:
  FOR temp = 500 TO 1000
    PULSOUT Servo pin,temp
    PAUSE 10
  NEXT
  FOR temp = 1000 TO 500
    PULSOUT Servo pin,temp
    PAUSE 10
  NEXT
  GOTO start
```

APPENDIX D

16F877 DATA SHEET



PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

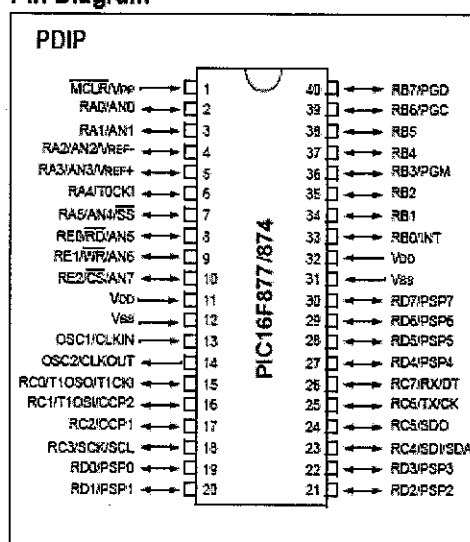
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F876
- PIC16F874
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
Up to 368 x 8 bytes of Data Memory (RAM)
Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC
oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM
technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two
pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature
ranges
- Low-power consumption:
 - < 0.6 mA typical @ 3V, 4 MHz
 - 20 µA typical @ 3V, 32 kHz
 - < 1 µA typical standby current

Pin Diagram

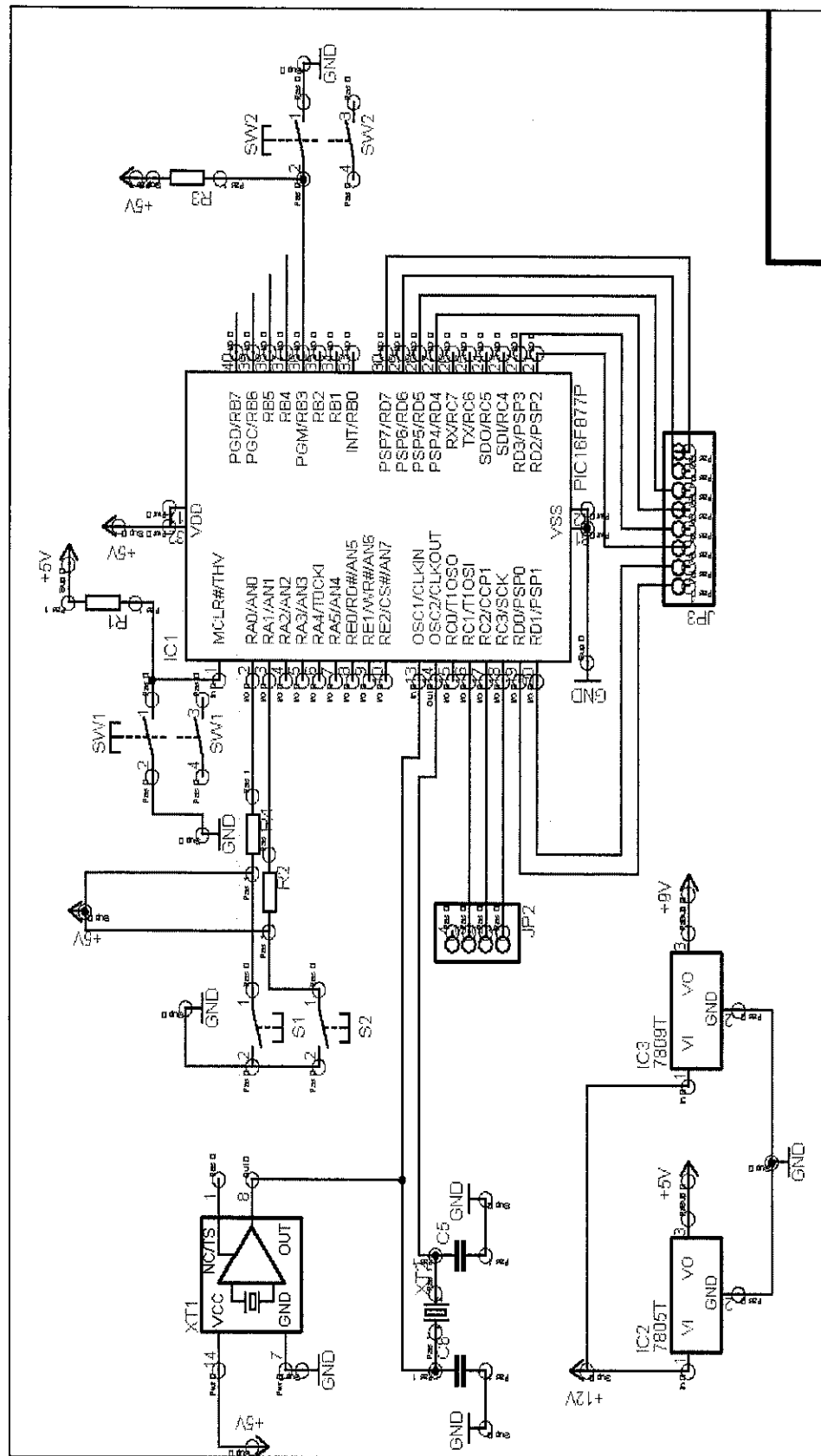


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler,
can be incremented during SLEEP via external
crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period
register, prescaler and postscaler
- Two Capture, Compare, PWM modules
 - Capture is 16-bit, max. resolution is 12.5 ns
 - Compare is 16-bit, max. resolution is 200 ns
 - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master
mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver
Transmitter (USART/SCI) with 9-bit address
detection
- Parallel Slave Port (PSP) 8-bits wide, with
external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for
Brown-out Reset (BOR)

APPENDIX E

MAIN CONTROLLER BOARD CIRCUIT



APPENDIX F

'MAIN.C' SOURCE CODES

```
#include <16F877.h>
#fuses XT,NOWDT,NOPROTECT, NOPUT, NOBROWNOUT,NOLVP
#use delay (clock=4000000)
#include <key_pad.c>
#include <LCD.C>

/*****
hexapod locomotion.c
combined programs
PIN_C0 = left servo
PIN_C1 = middle servo
PIN_C2 = right servo
*****/
*****Programmed by Oh Lay Shan (15/4/2006)*****/

main()
{
    int i;
    while(TRUE)
    {
        output_high(PIN_C3);
        output_high(PIN_C1);
        output_high(PIN_C2);
        delay_us(1500);
        output_low(PIN_C3);
        output_low(PIN_C1);
        output_low(PIN_C2);
        delay_ms (18);

        if(input(PIN_B3) == 0)
        {
            if (input(PIN_C5)==0)
                manual();
            else
                avoid();
        }
    }
}
```

APPENDIX G

'MANUAL.C' SOURCE CODE

```
void manual()
{
    int i;
    char k, e, b;
    int c;
    char d;

    while(TRUE)
    {
        i = 10;
        if (input(PIN_B3) == 0)
            return;
        if(k != 'N')
        {
            c=1;
            while (c==1)
            {
                char b;
                delay_ms(100);
                k = get_key();
                if(k != 'N')
                {
                    if (k == 'A' || k == 'B' || k == 'D')
                        e = k;
                    else
                        b = k;
                }
                switch(e)
                {
                    case 'A':
                    {
                        i=8;
                        d='5';
                        break;
                    }
                    case 'B':
                    {
                        i=12;
                        d='A';
                        break;
                    }
                    case 'D':
                    {
                        i=16;
                        d='F';
                        break;
                    }
                }
            }
            switch(b)
            {
```

```

        case '2':
        {
            forward(i);
            break;
        }
        case '3':
        {
            backward(i);
            break;
        }
        case '5':
        {
            turnleft(i);
            break;
        }
        case '6':
        {
            turnright(i);
            break;
        }
    }
    if (input(PIN_B3) == 0)
        return;
}
}
}
}
}

```

APPENDIX H

WALKING MOTION C CODES

//PROGRAM FOR BACKWARD MOTION

```
void backward (int i)
{
    int a;
    lcd_putc(i);
    for (a=0;a<i;a++)
    {
        output_high (PIN_C1);          // Frame 1
        delay_us(1600);
        output_low (PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high(PIN_C0);          // Frame 2
        output_high(PIN_C1);
        output_high(PIN_C2);
        delay_us(1400);
        output_low(PIN_C0);
        output_low(PIN_C2);
        delay_us(200);
        output_low(PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high (PIN_C1);          // Frame 3
        delay_us (1400);
        output_low (PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high(PIN_C0);          // Frame 4
        output_high(PIN_C1);
        output_high(PIN_C2);
        delay_us(1400);
        output_low(PIN_C1);
        delay_us(200);
        output_c (0x20);
        delay_ms (18);
    }
    return;
}
```

//PROGRAM FOR TURNING LEFT MOTION

```
void turnleft (int i)
{
    int a;
    lcd_putc(i);
    for (a=0;a<i;a++)
    {
        output_high (PIN_C1);           // Frame 1
        delay_us(1600);
        output_low (PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high(PIN_C0);           // Frame 2
        output_high(PIN_C1);
        output_high(PIN_C2);
        delay_us(1400);
        output_low(PIN_C0);
        delay_us(200);
        output_low(PIN_C2);
        output_low(PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high (PIN_C1);           // Frame 3
        delay_us (1400);
        output_low (PIN_C1);
        delay_ms (18);
    }
    for (a=0;a<i;a++)
    {
        output_high(PIN_C0);           // Frame 4
        output_high(PIN_C1);
        output_high(PIN_C2);
        delay_us(1400);
        output_low(PIN_C2);
        output_low(PIN_C1);
        delay_us(200);
        output_low(PIN_C0);
        delay_ms (18);
    }
    return;
}
```

//PROGRAM FOR TURNING RIGHT MOTION

```
void turnright (int i)
{
    int a;
    lcd_putc(i);
    for (a=0;a<i;a++)
```

```

{
    output_high (PIN_C1);           // Frame 1
    delay_us(1600);
    output_low (PIN_C1);
    delay_ms (18);
}
for (a=0;a<i;a++)
{
    output_high(PIN_C0);           // Frame 2
    output_high(PIN_C1);
    output_high(PIN_C2);
    delay_us(1400);
    output_low(PIN_C2);
    delay_us(200);
    output_low(PIN_C0);
    output_low(PIN_C1);
    delay_ms (18);
}
for (a=0;a<i;a++)
{
    output_high (PIN_C1);           // Frame 3
    delay_us (1400);
    output_low (PIN_C1);
    delay_ms (18);
}
for (a=0;a<i;a++)
{
    output_high(PIN_C0);           // Frame 4
    output_high(PIN_C1);
    output_high(PIN_C2);
    delay_us(1400);
    output_low(PIN_C0);
    output_low(PIN_C1);
    delay_us(200);
    output_low(PIN_C2);
    delay_ms (18);
}
return;
}

```

APPENDIX I

'AVOID.C' C CODES

```
void avoid()
{
    int i, j, position, k;

    while (TRUE)
    {
        i = 8;

        j = input_a() & 0x03;

        if (j==0x03)
        {
            forward (i);
        }

        if (j==0x01)    // left switch - move backward n turn right
        {
            for(k=0;k<4;k++)
            {
                backward(i);
            }
            for(k=0;k<4;k++)
            {
                turnright(i);
            }
        }

        if (j==0x02)    // right switch - move backward n turn left
        {
            for(k=0;k<4;k++)
            {
                backward(i);
            }
            for(k=0;k<4;k++)
            {
                turnleft(i);
            }
        }

        if (j==0x00)
        {
            for(k=0;k<5;k++)
            {
                backward(i);
            }
            for(k=0;k<6;k++)
            {
                turnright(i);
            }
        }
    }
}
```

```
    if (input(PIN_B3) == 0)
        return;
}
```


APPENDIX J

'LCD.C' C CODES

```

////////////////////////////////////
////          LCDD.C          ////
////          Driver for common LCD modules          ////
////          ////          ////
//// lcd_init() Must be called before any other function.  ////
////          ////          ////
//// lcd_putc(c) Will display c on the next position of the LCD.  ////
////          The following have special meaning:          ////
////          \f Clear display          ////
////          \n Go to start of second line          ////
////          \b Move back one position          ////
////          ////          ////
//// lcd_gotoxy(x,y) Set write position on LCD (upper left is 1,1)  ////
////          ////          ////
//// lcd_getc(x,y) Returns character at position x,y on LCD          ////
////          ////          ////
////////////////////////////////////

// As defined in the following structure the pin connection is as follows:
// RB0   Chip Enable (CE)
// RB1   Register Select (RS)
// RB2   Read/Write* (R/W*)
// RB4   Data Bit 4 (DB4)
// RB5   Data Bit 5 (DB5)
// RB6   Data Bit 6 (DB6)
// RB7   Data Bit 7 (DB7)
//
// LCD pins DB0-DB3 are not used and PIC's RB3 is not used.

// Un-comment the following define to use port B
#define use_portb_lcd TRUE

struct lcd_pin_map {          // This structure is overlayed
    BOOLEAN enable;          // on to an I/O port to gain
    BOOLEAN rs;              // access to the LCD pins.
    BOOLEAN rw;              // The bits are allocated from
    BOOLEAN unused;          // low order up. ENABLE will
    int data : 4;            // be pin B0.
} lcd;

#if defined(__PCH__)
#if defined use_portb_lcd
    #byte lcd = 0xF81          // This puts the entire structure
#else
    #byte lcd = 0xF83          // This puts the entire structure
#endif
#endif
#if defined use_portb_lcd

```

```

    #byte lcd = 6          // on to port B (at address 6)
#else
    #byte lcd = 8          // on to port D (at address 8)
#endif
#endif

#if defined use_portb_lcd
    #define set_tris_lcd(x) set_tris_b(x)
#else
    #define set_tris_lcd(x) set_tris_d(x)
#endif

#define lcd_type 2        // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40 // LCD RAM address for the second line

BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
    // These bytes need to be sent to the LCD
    // to start it up.

    // The following are used for setting
    // the I/O port direction register.

struct lcd_pin_map const LCD_WRITE = {0,0,0,0,0}; // For write mode all pins are
out
struct lcd_pin_map const LCD_READ = {0,0,0,0,15}; // For read mode data pins are
in

BYTE lcd_read_byte() {
    BYTE low,high;
    set_tris_lcd(LCD_READ);
    lcd.rw = 1;
    delay_cycles(1);
    lcd.enable = 1;
    delay_cycles(1);
    high = lcd.data;
    lcd.enable = 0;
    delay_cycles(1);
    lcd.enable = 1;
    delay_us(1);
    low = lcd.data;
    lcd.enable = 0;
    set_tris_lcd(LCD_WRITE);
    return( (high<<4) | low);
}

void lcd_send_nibble( BYTE n ) {
    lcd.data = n;
    delay_cycles(1);
    lcd.enable = 1;
    delay_us(2);
    lcd.enable = 0;

```

```

}

void lcd_send_byte( BYTE address, BYTE n ) {

    lcd.rs = 0;
    while ( bit_test(lcd_read_byte(),7) );
    lcd.rs = address;
    delay_cycles(1);
    lcd.rw = 0;
    delay_cycles(1);
    lcd.enable = 0;
    lcd_send_nibble(n >> 4);
    lcd_send_nibble(n & 0xf);
}

void lcd_init() {
    BYTE i;
    set_tris_lcd(LCD_WRITE);
    lcd.rs = 0;
    lcd.rw = 0;
    lcd.enable = 0;
    delay_ms(15);
    for(i=1;i<=3;++i) {
        lcd_send_nibble(3);
        delay_ms(5);
    }
    lcd_send_nibble(2);
    for(i=0;i<=3;++i)
        lcd_send_byte(0,LCD_INIT_STRING[i]);
}

void lcd_gotoxy( BYTE x, BYTE y) {
    BYTE address;

    if(y!=1)
        address=lcd_line_two;
    else
        address=0;
    address+=x-1;
    lcd_send_byte(0,0x80 | address);
}

void lcd_putc( char c) {
    switch (c) {
        case '\f' : lcd_send_byte(0,1);
                    delay_ms(2);
                    break;
        case '\n' : lcd_gotoxy(1,2);      break;
        case '\b' : lcd_send_byte(0,0x10); break;
        default  : lcd_send_byte(1,c);    break;
    }
}

char lcd_getc( BYTE x, BYTE y) {

```

```
char value;

lcd_gotoxy(x,y);
while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
lcd.rs=1;
value = lcd_read_byte();
lcd.rs=0;
return(value);
}
```

APPENDIX K

'KEYPAD.C' C CODES

```
//#byte port_d = 0x08
char get_key(void)
{
    char t;
    while (1) {

        output_d(input_d() | 0xFF);    /* set RD4 to low to scan the first row */

        output_bit(PIN_D4,0);

        if (input(PIN_D0) ==0){
            delay_us(10);

            while(input(PIN_D0) ==0)
            {
            }

            return 'A';    /* return the ASCII code of A */
        }

        if (input(PIN_D1) ==0) {
            delay_ms(10);
            while(input(PIN_D1) ==0)
            {
            }

            return '7';    /* return the ASCII code of 7 */
        }

        if (input(PIN_D2) ==0) {
            delay_ms(10);
            while(input(PIN_D2) ==0)
            {
            }

            return '4';    /* return the ASCII code of 4 */
        }

        if (input(PIN_D3) ==0) {
            delay_ms(10);
            while(input(PIN_D3) ==0)
            {
            }

            return '1';    /* return the ASCII code of 1 */
        }

        output_d(input_d() | 0xFF);    /* set RD5 to low to scan the second row */
    }
}
```

```

        output_bit(PIN_D5,0);
        if (input(PIN_D0) ==0) {
            delay_ms(10);
while(input(PIN_D0) ==0)
{
}
            return '0';    /* return the ASCII code of 0 */
        }

        if (input(PIN_D1) ==0) {
            delay_ms(10);
while(input(PIN_D1) ==0)
{
}
            return '8';    /* return the ASCII code of 8 */
        }

        if (input(PIN_D2) ==0) {
            delay_ms(10);
while(input(PIN_D2) ==0)
{
}
            return '5';    /* return the ASCII code of 5 */
        }

        if (input(PIN_D3) ==0) {
            delay_ms(10);
while(input(PIN_D3) ==0)
{
}
            return '2';    /* return the ASCII code of 2 */
        }

output_d(input_d() | 0xFF);    /* set RD6 to low to scan the third row */
        output_bit(PIN_D6,0);
        if (input(PIN_D0) ==0) {
            delay_ms(10);
while(input(PIN_D0) ==0)
{
}
            return 'B';    /* return the ASCII code of B */
        }

        if (input(PIN_D1) ==0) {
            delay_ms(10);
while(input(PIN_D1) ==0)
{
}
            return '9';    /* return the ASCII code of 9 */
        }

        if (input(PIN_D2) ==0) {
            delay_ms(10);
while(input(PIN_D2) ==0)
{
}

```

```

        return '6';    /* return the ASCII code of 6 */
    }

    if (input(PIN_D3) ==0) {
        delay_ms(10);
        while(input(PIN_D3) ==0)
        {
        }
        return '3';    /* return the ASCII code of 3 */
    }

output_d(input_d() | 0xFF);    /* set RD7 to low to scan the fourth row */
    output_bit(PIN_D7,0);
    if (input(PIN_D0) ==0) {
        delay_ms(10);
        while(input(PIN_D0) ==0)
        {
        }
        return 'C';    /* return the ASCII code of C */
    }

    if (input(PIN_D1) ==0) {
        delay_ms(10);
        while(input(PIN_D1) ==0)
        {
        }
        return 'D';    /* return the ASCII code of D */
    }

    if (input(PIN_D2) ==0) {
        delay_ms(10);
        while(input(PIN_D2) ==0)
        {
        }
        return 'E';    /* return the ASCII code of E */
    }

    if (input(PIN_D3) ==0) {
        delay_ms(10);
        while(input(PIN_D3) ==0)
        {
        }
        return 'F';    /* return the ASCII code of F */
    }
else
return 'N';
    }
}

```